
ARRL AMATEUR RADIO

**10TH COMPUTER
NETWORKING
CONFERENCE**



SAN JOSE, CALIFORNIA

SEPTEMBER 27-29, 1991



Copyright 1991 by

The American Radio Relay League, Inc.

International Copyright secured

All rights reserved. No part of this work may be reproduced, in any form except by written permission of the publisher. All rights of translation are reserved.

Printing History
Published 1991 ARRL
TAPR edition / 1995

Printed in USA

ISBN: 0-87259-359-2

Second Edition

Foreword

The American Radio Relay League takes pride in publishing papers for this Tenth ARRL Amateur Radio Computer Networking Conference.

This year's papers represent a new level in concepts for further development of the amateur packet radio network. There are papers discussing mail transfer, data compression, modems, higher transmission speeds, spectrum efficiency, digital signal processing and a new system called Clover II.

As in previous conferences, all papers in these proceedings are unedited and solely the work of the authors.

David Sumner, **K1ZZ**
Executive Vice President

Newington, Connecticut
September, 1991

Table of Contents

extended Mail Transfer Protocol (XMTP) by J. Gordon Beattie, Jr., N2DSY, Andrew R. Funk, KB7UV and Frank Warren, KB4CYC	I
A Full-Duplex 56 kb/s CSMA/CD Packet Radio Repeater System by Mike Cheponis, K3MC and Lars Karlsson, AA6IW	11
Characterization of PACSAT-1 Traffic via Downlink Monitoring by Robert J. Diersing, N5AHD	16
Experimental Study of Shannon-Fano, Huffman, Lempel-Ziv-Welch and Other Lossless Algorithms by D. Dueck and W. Kinsner, VE4WK	23
Recent Hubmaster Networking Progress in Northern California by Glenn Elmore, N6GN and Kevin Rowett, N6RCE	32
Distributed Directory Services for the Amateur Packet Radio Network by Andrew Funk, KB7UV	45
Specification of the AVC-R-ISA Mac Layer Protocol by A. Giordano, I1TD, A. Imovilli, IW1PWV, C. Nobile, IW1QAP, G. Pederiva, IW1QAN and S. Zappatore, IW1PTR	52
Spectral Efficiency Considerations for Packet Radio by Phil Karn, KA9Q	62
Lossless Data Compression Algorithms for Packet Radio by W. Kinsner, VE4WK	67
LZW Compression of Interactive Network Traffic by Anders Klemets, SMØRGV	76
Proposed Design and Strategy for a Radio Direction Finding Network Using Doppler Antennas, Packet, Spread Spectrum, and Transmitter Signatures by Digital Signal Processing by Andrew J. Korsak, Ph.D., VE3FZK/W6	82

Design and Implementation of CELP Speech Processing System Using TMS320C30 by A. Langi, VE4ARM and W. Kinsner, VE4WK	87
Digital Networking With the WA4DSY Modem - Adjacent Channel and Co-Channel Frequency Reuse Considerations by Ian McEachern, VE3PFH	94
The 56 kb/s Modem as a Network Building Block: Some Design Considerations by Barry McLarnon, VE3JF	100
Multi-Drop KISS Operation by Karl Medcalf, WK5M	109
The Shape of Bits to Come by James Miller, BSc, G3RUH	132
The Ottawa Packet Interface (PI): A Synchronous Serial PC Interface for Medium Speed Packet Radio by Dave Perry, VE3IFB	121
Clover II: A Technical Overview by Raymond C. Petit, W7GHM	125
Improving the Packet Mail Transfer System by Brian B. Riley, KA2BQE	130
GUI Packet by Keith Sproul, WU2Z and Mark Sproul, KB2ICI	137
NOS Command Set Reference by Ian Wade, G3NRW	148
Higher Speed Amateur Packet Radio Using the Apple Macintosh Computer by Doug Yuill, VE3OCU	154

extended Mail Transfer Protocol (XMTP)

J. Gordon **Beattie**, Jr., **N2DSY**
Andrew R. Funk, **KB7UV**
Frank Warren, **KB4CYC**

The Radio Amateur Telecommunications Society
206 North Vivyen Street
Bergenfield, New Jersey 07621
Telephone: **+1.201.387.8896**

ABSTRACT

The amount of store-and-forward traffic in the Amateur Radio Packet Network has increased to the point where significant optimization of the message forwarding scheme for packet bulletin board and other servers is required. The purpose of this paper is to present an enhanced message forwarding system which we call the "**eXtended Mail Transfer Protocol**" or XMTP which addresses this requirement. Further, an overall client/server model is included as a possible implementation enhancement to systems which plan to implement this protocol.

INTRODUCTION

In the current Amateur Packet Radio Network there are servers which are usually based on MS-DOS computers which use the ARRL AX.25 Link Layer Protocol to move store-and-forward messages across the network. The ROSE X.25 networking protocol is used in the authors' network and for the examples in this paper, but any of the other common networking protocols could be substituted.

The basic application environment architecture for the evolving **ROSE** environment depends on the ROSE X.25 and the AX.25 protocol to convey bits and bytes. Over these connections (or "**path**" for all you connectionless folks), the **ROSE** environment uses a client/server protocol called "Serialized Transaction Interface **Protocol**" or S-TIP to provide a remote operations invoker/responder facility. One of the "**users**" of S-TIP is XMTP. XMTP depends on the underlying services of S-TIP to provide the facilities of a transaction monitor. These functions include signaling and verification of transaction completeness, confirmation delivery if required by the S-TIP user, transfer syntax signaling including compression, application addressing, and application capability negotiation between systems.

S-TIP will be described in a future paper, but it is based on some of the concepts and capabilities of CCITT X.219/X.229 Remote Operations Services, Sun **Microsystems'** Remote Procedure Call (RPC) and the communications facilities of the "**MINIX**" operating system.

The role of XMTP is to provide a uniform and extensible platform for the movement of store-and-forward traffic. XMTP can be implemented using the current message header data, but the use of additional

header elements as found in either the CCITT X.400 Message Handling System and RFC-822 is strongly recommended. The convergence of these two protocols as outlined in RFC-987, RFC-1138 and RFC-1148 would facilitate the automatic interface of a great body of existing OSI and Internet software and systems. Gateways are only a part of the benefit. The software available would improve the **user** interface, add multiple personal mail delivery and conference services, allow for mixed graphical, voice and text messages, as well as other interesting features.

XMTP FEATURES

XMTP provides for the rapid transmission of store-and-forward messages between systems. The data flow **diagrammes** included in this paper illustrate the connection-oriented process, but the same basic procedure could be used in connectionless or "**multicast**" environments.

The current dialogue between store-and-forward systems causes systems to send and wait for a response. In most networks, **the** network transit delay is high and this wastes time. In order to reduce this loss of time, XMTP optimizes the dialogue by reducing the number of times that a transaction "holds-up" the data flow during a forwarding cycle. The basic changes that XMTP brings to the **store-and-forward** message environment are:

1. The pre-registration of the capabilities of a system eliminates the need to exchange them before each data transfer. The current "SSID" or Smart System ID which is sent in brackets "[...]" at the start of every session is now eliminated except when such a message is received from the other system. This allows for backwards compatibility as well as for the recovery of peer system capability information after a crash or change.
2. The headers of all mail queued for forwarding are sent in a single exchange. Currently, systems send a short header **including** the Bulletin or Message ID and wait for a "**go/no-go**" response.- After receipt of a "**go**", the system sends the message and then waits for an acknowledgement before sending the next header.
3. XMTP supports simultaneous bidirectional forwarding. This reduces the store-and-forward transfer time significantly by filling both directions of the channel at the same time.

Some have suggested that all messages should be compressed into a single file and then transmitted. This could lead to very large transfer files and possibly cause the less than timely delivery of a particular message. XMTP submits each message to **S-TIP** as a separate transaction. Each transaction is compressed, transferred, and decompressed by the underlying service provided by S-TIP. S-TIP does not control this process, but simply acts as directed by XMTP. As such, if future changes to XMTP include the transfer of **multiple** messages in a single compressed package, then S-TIP will transparently handle the requirement.

XMTP FLOW DIAGRAMMES

XMTP supports three basic modes of operation: Simple Forwarding, Polled Forwarding and Duplex Forwarding. This section outlines the data flow process for each mode. The flow diagrammes along with a general architecture figure are at the end of this paper.

SIMPLE FORWARDING CASE

In the Simple forwarding case, the connection is established and then a Mail-Q-Event-Report is sent to the receiving system. This report summarizes all mail traffic for that system. It provides information which allows the receiving system to determine the priority which it will use to receive these messages, as well as if there is sufficient space and if the message is a duplicate. The message elements are outlined in the **"XMTP DATA ELEMENTS"** section of this paper. The receiving system then sets the appropriate status for each message on the sending system by sending a Mail-Q-Set-Status message. The sending system then starts to send each message as Mail-Create-Requests without waiting for individual acknowledgements. It should be noted at this point that what the sending system is doing is creating a message just like the one it has, on the receiving system. The receiving system can then send Mail-Log-Event-Report-Request messages reflecting the successful receipt of one or more messages. This may be repeated as additional messages are received.

SIMPLE POLLING CASE

In the Simple polling case, the connection is established and then a Mail-Q-Set-Request is sent to trigger the **"sending"** system into providing a list of queued messages. The sending system then sends a Mail-Q-Event-Report to the receiving system as was done in the Simple Forwarding case. The receiving system then sets the appropriate status for each message on the sending system by sending a Mail-Q-Set-Status message. The sending system then starts to send each message as Mail-Create-Requests without waiting for individual acknowledgements. It should be noted at this point that what the sending system is doing is creating a message just like the one it has, on the receiving system. The receiving system can then send Mail-Log-Event-Report-Request messages reflecting the successful receipt of one or more messages. This may be repeated as additional messages are received.

DUPLEX FORWARDING CASE

In the Duplex forwarding case, the flow is the same as in either or both of the other two cases, but data is allowed to flow in each direction at once.

ACKNOWLEDGEMENTS

The authors wish to thank all the users of the **ROServer/PRMBS** software package as well as **it's** author, Brian Riley, **KA2BQE**, for their work and comments on the protocol and the research that went into this paper. The authors also would like to thank Nancy **Beattie**, **N2FWI**, Ted Beauchamp, **KA2USU**, David Elliot, **KD6TH**, Tom Moulton, **W2VY**, Bob Nelson, **KB1BD**, Buck Rogers, **K4ABT**, Don Rotolo, **N2IRZ**, and Bill Slack, **NX2P** for their research and observations.

XMTP DATA ELEMENTS

[S-TIP-.../XMTP-...]

This is the Smart System ID message element included in the "[]" message exchange performed after connection time. This signals each side that S-TIP and **XMTP** Services are available and that the Application Manager Managed Object is present. Specific selection of a suitable string is needed.

S-TIP Header

The S-TIP Header is based on the OSI model and uses the connectionless, Unit-Data Services to implement the protocol. Each Protocol Data Unit (PDU) is sent to the service interface by an application, and then delivered to a peer application entity. Some minimal state information about the PDU is maintained by the S-TIP provider. Some changes and enhancements are currently being implemented. Please contact the authors for the revised **format**.

```
S-TIP Header ::= SEQ OF {  
    Network- Source Address, Dest Address, Header Chk,  
              Data Length  
    Transport- Source Address, Dest Address, Checksum  
    Session- Source Address, Dest Address  
    Presentation- Source Address, Dest Address, PCI  
    Application Context Name  
    Operation Code /* Such as mail message type */  
    Mode /* Best Effort, Atomic, etc. */  
    InvokeID /* This is a transaction number */  
}
```

Mail-Q-Event-Report

This message is a notification sent by a system to signal the other system of the availability of mail. This message may **be** sent any time during a communication. The format is as follows:

```
Mail-Q-Event-Report ::= SEQ OF {  
    S-TIP Header  
    Object Class = Mail-Q-Summary-Record  
    Object Instance = SystemName&RecordID  
    Operation Time = UTC  
    Operation Type = Mail-Q-Event-Report  
    Operation Data = SET OF Mail-Q-Record  
}
```

```

Mail-Q-Record ::= SEQ OF {
    Object Class = Mail-Q-Record
    Object Instance = Mail-Q-RecordID
    Date/Time = UTC
    Hold Date/Time = UTC
    Mail Type ::= CHOICE { P, T, B, 0, . . . }
    Status ::= CHOICE { Y, N, D, K, . . . }
    Q-Status ::= CHOICE { G, R, H, . . . }
    Size = Uncompressed Byte Count
    To = FullyQualifiedAddressee
    From = FullyQualifiedAddressee
    Subject = OctetString
    Path = SEQ OF { Mail-Q-RecordID }
}

```

```

Mail-Q-RecordID ::= SEQ OF {
    Message Number
    "@@
    SystemID
}

```

```

FullyQualifiedAddressee ::= SEQ OF (
    CHOICE { Callsign | ApplicationName )
    Device ::= Callsign + Unique Identifier
    Organization ::= OctetString /* the # stuff */
    Locality ::= CHOICE { State | Province | etc. }
    Country ::= ISO3166-Alpha-2
}

```

Subject = **OctetString**

The default values for Q-Status and Hold Date/Time are "**H**" and "**0**".
Q-Status values are: G for Get, R for Remove and H for Hold.

Mail-Q-Set-Request

This message signals the other system to alter the status of Mail-Q-Record Q-Status attribute. This change can cause a mail message to be held until a later time (H), removed (R), or retrieved (**G**). The default values for Q-Status and Hold Date/Time are "**H**" and "**0**". This message may be sent anytime during a communication. The format is as follows:

```

Mail-Q-Set-Request ::= SEQ OF (
    S-TIP Header
    Object Class = Mail-Q-Record
    Object Instance = SystemName
    Operation Type = Mail-Q-Set-Request
    Operation Data = SET OF Mail-Q-Status
}

```

```

Mail-Q-Set-Status ::= SEQ OF {
    Object Class = Mail-Q-Record
    Object Instance = Mail-Q-RecordID
    Hold Date/Time = UTC
    Q-Status ::= CHOICE { G, R, H, . . . }
}

```

Mail-Create-Request

This message signals the other system to create a new mail object. The default values for Q-Status and Hold Date/Time are "H" and "0". This message may be sent anytime during a communication. The format is as follows:

```
Mail-Create-Request ::= SEQ OF {  
    S-TIP Header  
    Object Class = Mail-Record  
    Object Instance = Mail-Q-RecordID  
    Operation Time = UTC  
    Operation Type = Mail-Create-Request  
    Operation Data = SEQ OF {  
        Mail-Q-Record  
        Message-Body  
    }  
}
```

The default values for Q-Status and Hold Date/Time are "H" and "0". The Object Instance (Record ID) of the Mail-Create-Request is the message number used on the local system. The sender will always use its local message number. The receiver will replace the received message number with its own local value.

Mail-Q-Set-Request

This message signals the other **system** to send the list of Mail-Q-Records. This message may be sent anytime during a communication. The format is as follows:

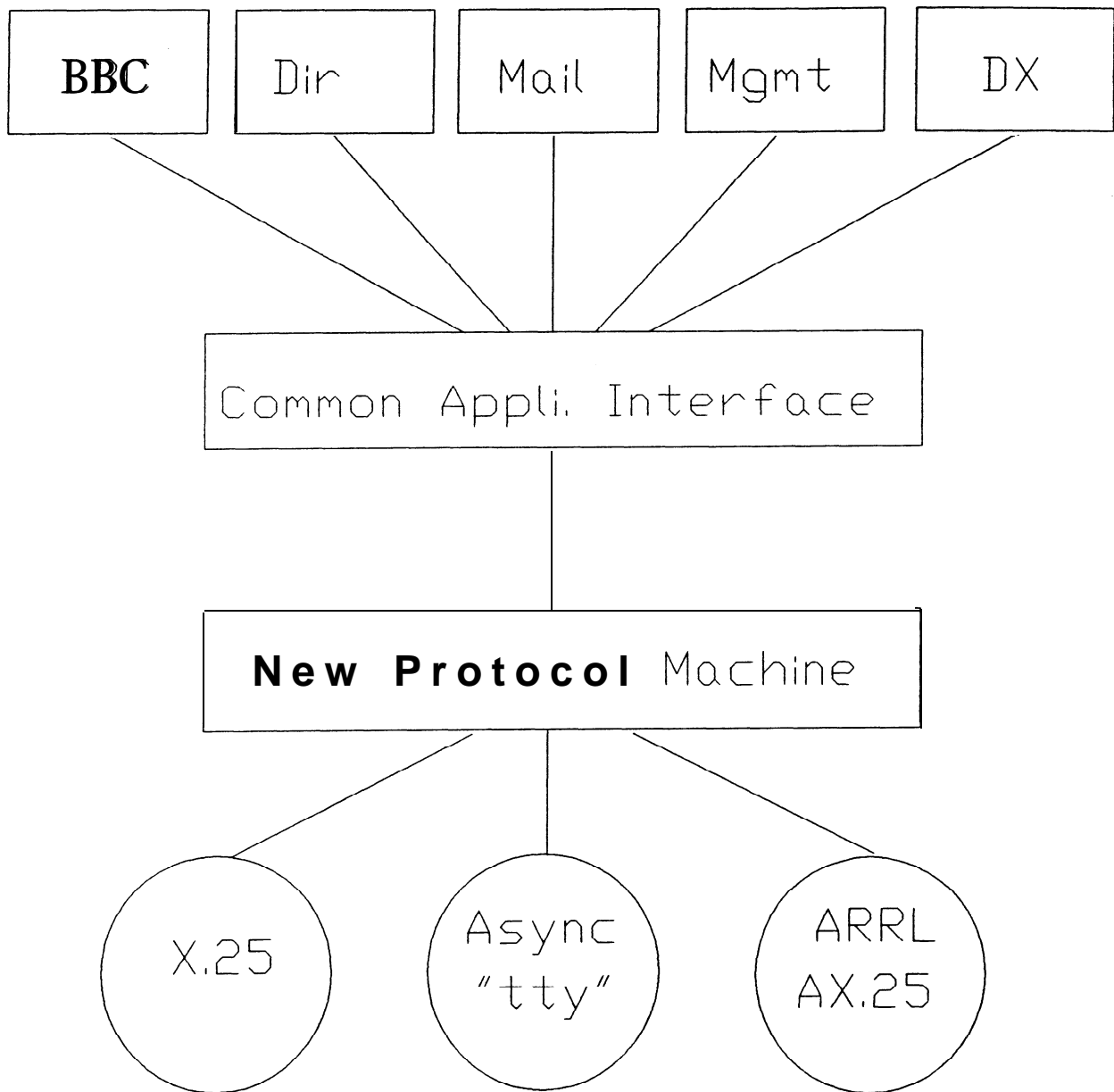
```
Mail-Q-Set-Request ::= SEQ OF {  
    S-TIP Header  
    Object Class = Mail-Q-Record  
    Object Instance = SystemName  
    Operation Type = Mail-Q-Set-Request  
    Operation Data = SET OF { Mail-Q-RecordID }  
}
```

Mail-Q-Set-Response

This message is the response to the request for a list of Mail-Q-Records. The format is as follows:

```
Mail-Q-Set-Response ::= SEQ OF {  
    S-TIP Header  
    Object Class = Mail-Q-Summary-Record  
    Object Instance = SystemName&RecordID  
    Operation Time = UTC  
    Operation Type = Mail-Q-Event-Report  
    Operation Data = SET OF Mail-Q-Record  
}
```

New Protocol Implementation



file:stack

eXtended Mail Transfer Protocol (XMTP)

Simple Forwarding Case

System A		System B
>	Connection Request	>
<	Connection Accept	<
>	Mail-Q-Event-Report	>
<	Mail-Q-Set-Request	<
>	Mail-Create-Request (Msg_1)	>
>	Mail-Create-Request (Msg_2)	>
>	Mail-Create-Request (Msg_3)	>
>	Mail-Create-Request (Msg_4)	>
<	Mail-Log-Event-Report-Request (1-4)	<

extended Mail Transfer Protocol (XMTP)

Simple Polling Case

System A		System B
<	Connection Request	<
>	Connection Accept	>
c	Mail-Q-Set-Request	<
>	Mail-Q-Event-Report	>
<	Mail-Q-Set-Request	<
>	Mail-Create-Request (Msg_1)	>
>	Mail-Create-Request (Msg_2)	>
>	Mail-Create-Request (Msg_3)	>
>	Mail-Create-Request (Msg_4)	>
<	Mail-Log-Event-Report (1-4)	c

eXtended Mail Transfer Protocol (XMTP)

Duplex Forwarding Case

System A		System B
>	Connection Request	>
<	Connection Accept	<
>	Mail-Q-Event-Report	>
<	Mail-Q-Event-Report	<
<	Mail-Q-Set-Request	<
>	Mail-Q-Set-Request	>
>	Mail-Create-Request (Msg_1)	>
<	Mail-Create-Request (Msg_A)	<
>	Mail-Create-Request (Msg_2)	>
<	Mail-Create-Request (Msg_B)	<
>	Mail-Log-Event-Report (A-B)	>
>	Mail-Create-Request (Msg_3)	>
>	Mail-Create-Request (Msg_4)	>
<	Mail-Log-Event-Report (1-4)	<

A Full-Duplex 56kb/s CSMA/CD Packet Radio Repeater System

Mike Cheponis, K3MC
Lars Karlsson, AA6IW

ABSTRACT

There are no **true** Carrier Sense, Multiple Access with Collision Detect (CSMA/CD) systems operational in amateur radio. **Full-duplex** systems **currently** in use are actually **CSMA/CA** - that is, CSMA with Collision Avoidance. We describe here a system that accomplishes full collision detection with little additional system complexity compared with an ordinary full-duplex system.

1. Overview

The idea of a full-duplex **bit**-regenerative packet repeater is not new. [1-5] Because of the **well**-known disadvantages of **CSMA**, schemes like full-duplex repeaters are suggested because they eliminate the hidden transmitter problem. Although there are other ways to eliminate hidden terminals, for example, busy tone and polling schemes [6-7], full-duplex provides the capability for the transmitting station to *listen while transmitting*. This allows the station to determine if the bytes being sent are the same as the bytes being received. If so, then all is well. If not, then a collision is occurring, and transmission should immediately cease.

By taking advantage of this capability to quickly detect collisions, an extremely high performance packet radio system can be constructed, with little additional system complexity compared with an ordinary full-duplex repeater.

2. Our Approach

We have used off-the-shelf hardware to augment a simple digital regenerative scheme with some intelligence. This scheme not only allows repeater operation, and all of the benefits of full duplex, but also permits trivial **connection** with other **KA9Q TCP/IP** networks.

The code that runs in the user node is the same as the code that runs in the repeater. We did this to reduce the coding effort, but also it is a nice fallout of this scheme. Basically, the code is the same as that for a KISS TNC [8], with a few straightforward extensions. These extensions are:

- A function to store the AX.25 link layer id (callsign + SSID) inside the digital hardware.
- A function that compares **byte-for-byte** in real-time the incoming (or repeated, in the case of the repeater) packet with the outgoing packet, and **interrupts** transmission in case the bytes don't match.

- A function to compare the incoming packet's link layer ID to the stored link layer ID, and to insert that packet at the head of a priority queue, thereby ensuring that the connected computer gets important packets, and avoids overload of the connected computer.

End user TCP/IP routing is accomplished by first an "arp add" like **this**:

```
arp add ax25 aa6iw.ampr.org aa6iw-0
```

What this tells the end user's TCP/IP setup is that all packets for the IP name "aa6iw.ampr.org" are to be addressed using a link address of "aa6iw-0".

An end user that wishes to be connected to the rest of the local internet would need the following "route add" statement:

```
route add default rp0 aa6iw.ampr.org
```

which would cause all IP datagrams that are not explicitly routed to go via the aa6iw.ampr.org gateway.

For each station that would use this repeater, a separate "route add" statement would be required. So, for example, if hs.k3mc.ampr.org uses the repeater, and he knows that aa4cg.ampr.org also uses the repeater, hs.k3mc.ampr.org needs:

```
route add aa4cg.ampr.org rp0
```

¹Note that aa6iw.ampr.org is in fact the gateway computer that is attached to our repeater node. It is connected via an ordinary RS-232 cable with KISS protocol.

in his KA9Q NET.EXE initialization². Similarly, aa4cg.ampr.org needs the following line for his NET.EXE initialization:

```
route add hs.k3mc.ampr.org rp0
```

Additional repeater users need to have individual "route add" statements for every user of the repeater. Users on the other networks that are accessed via the gateway computer, however, are handled by the default routing.

3. Hardware

The heart of the repeater is, of course, the Dale Heatherington, WA4DSY 56 kilobit modem[9]. This truly beautiful design is serving as the workhorse for many advanced packet systems. The Georgia Radio Amateur Packet Enthusiasts Society (GRAPES) makes this board set and a parts kits available to amateurs at nominal cost³.

The digital portion uses the standard Kantronics Data Engine. This V40-based controller has all the power needed to handle this application. In particular, the Data Engine has a Zilog 8530 SCC dual-channel serial controller chip. In our implementation, one channel (the over-the-air one) is DMA-driven, and

²"rp0" is just an arbitrary designator that associates a NET.EXE interface with a name: for this example, it means "Repeater 0".

³Doug Drye, KD4NC, and a band of other enthusiasts makes the distribution of Dale's radio possible.

the other is the high-speed KISS link back to a host **computer**⁴.

The end user stations and the repeater naturally differ on RF requirements.

The end user stations use a simple MMIC-based 906.65 to 29.05 MHz downconverter. The transmitting upconverter is a Hamtronics XV-4. The user antennas are on a single boom: The 430 antenna is vertically polarized, and the 900 MHz antenna is horizontally polarized. These two antennas are in a "**cross**" configuration, similar to typical satellite antennas. The number of elements can be varied; the particular user station configuration depends upon required BER for satisfactory repeater system performance. See Figure 1. The single-boom dual-band antenna is fed with a single cable, and split inside the shack.

The repeater site uses a simple downconverter on receive, and an MMIC-based upconverter for transmit. A Power Amplifier boosts output to 10 to 50 watts. Omnidirectional antennas are used at the repeater site, and are fed with a single feedline, like the user case.

As you can see **from** Figure 2, the repeater needs a simple AND/OR gate addition. The functions of these parts **is** simple. The OR gate merely allows the digital card to hear what it is transmitting (remember, we are

running identical code in the user nodes and the repeater node)? When we are doing this, we must ignore what would normally be received, and this is the function of the AND gate. Note that when the repeater wants to talk, everybody else listens!

4. Conclusion

This scheme reduces the "**window** of vulnerability" of ordinary **CSMA** and **CSMA/CA** systems to very low levels. It eliminates both hidden and exposed terminals. Since collisions don't cost much in lost channel time, we can send larger packets and reduce per-packet overhead to quite low levels.

We look forward to continuing to work on this system, and to gather quantitative data to assess the **CSMA/CD** performance.

You may contact us at these addresses:

Mike:

k3mc@k3mc.#nocal.ca.usa.na (BBS)

k3mc@tandem.com (Internet)

k3mc@k3mc.ampr.org (Amprnet)

Lars:

aa6iw@k3mc.#nocal.ca.usa.na (BBS)

aa6iw@aa6iw.ampr.org (Amprnet)

⁴**Please** note that our design **decision** to use a standard **KISS** serial link means that **we** do not depend on a particular implementation of NET.EXE on a particular machine; IBM PCs, Macintoshes, **Amigas**, etc., can all use KISS Interfaces.

⁵**This** would happen when the RS-232 network connection requires the repeater to originate a **message onto the repeater system**.

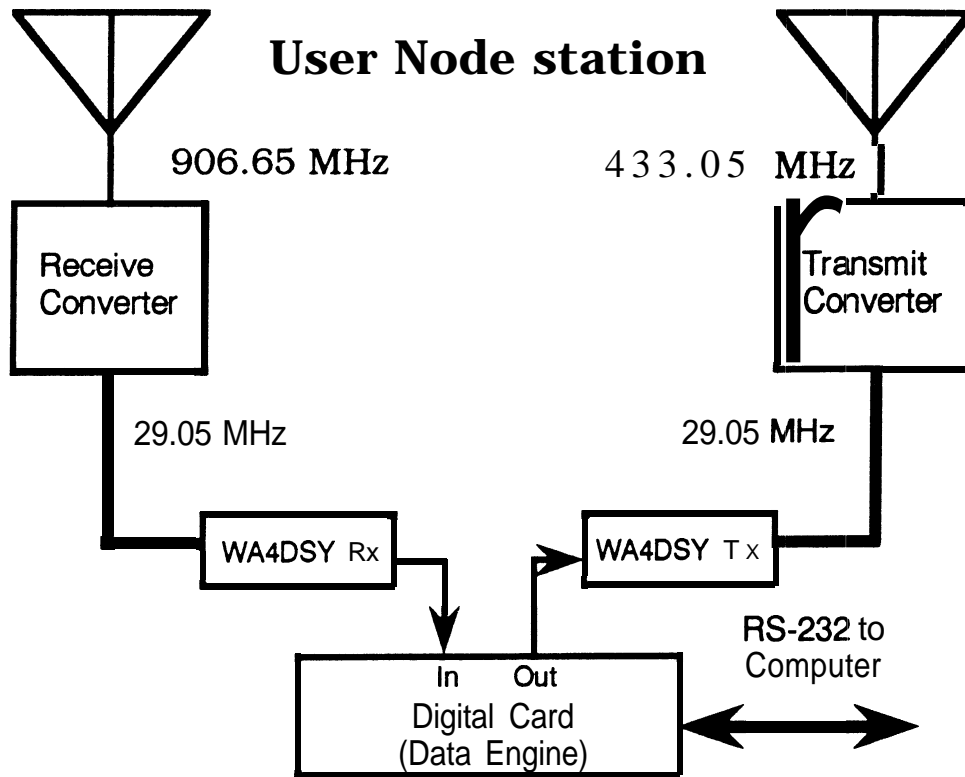


Figure 1

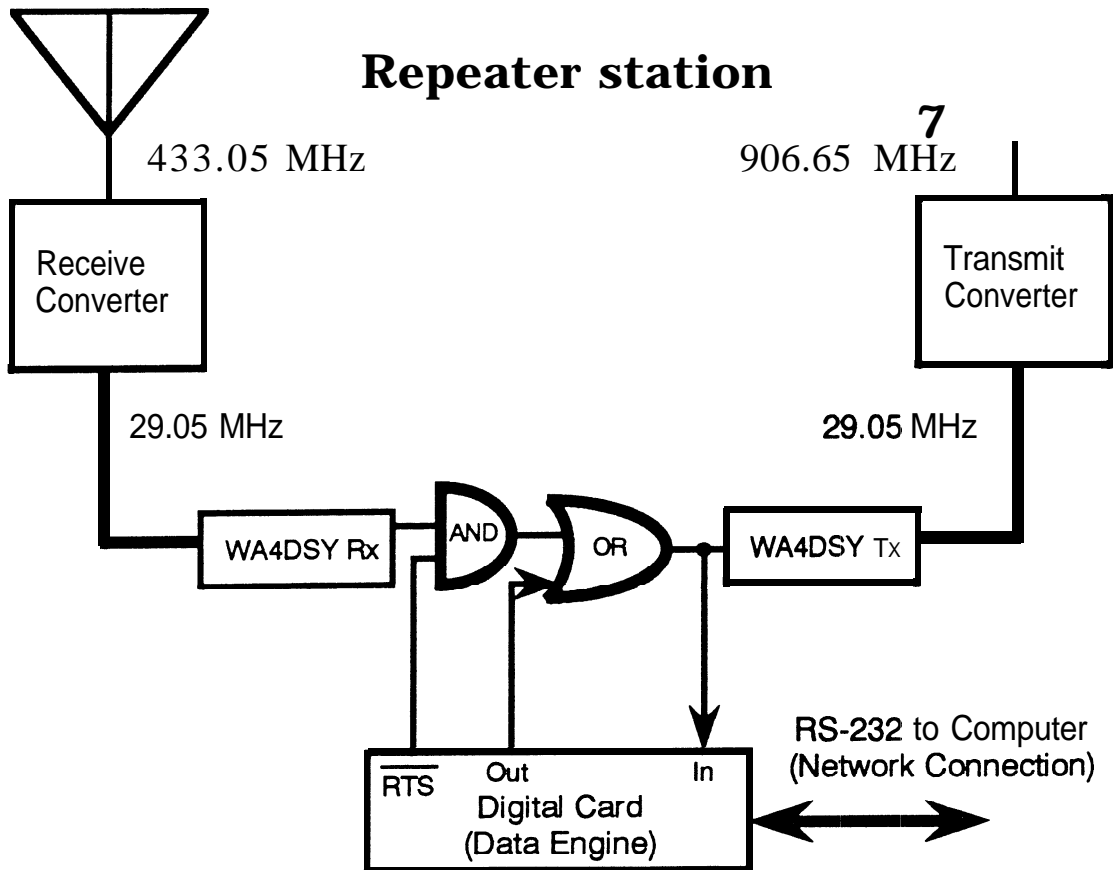


Figure 2

References

- [1] **Magnuski**, Hank **KA6M** "On the Care and Feeding of Your Packet Repeater." First **ARRL Amateur Radio Computer Networking Conference**, Volume 2 p9 [1981]
- [2] Fox, **Terry WB4JFI** 'RF, RF Where is My High Speed RF?" 5th **ARRL Amateur Radio Computer Networking Conference** p 5.1 [1986]
- [3] Finch, Robert **N6CXB** and **Avent**, Scott **N6BGW** "A Duplex Packet Radio Repeater Approach to Layer One Efficiency, " 6th **ARRL Amateur Radio Computer Networking Conference** p 47 [1987]
- [4] **Karn**, Phil **KA9Q** "A High Performance, Collision-Free Packet Radio Network" 6th **ARRL Amateur Radio Computer Networking Conference** p 86 [1987]
- [5] **Avent**, Scott **N6BGW** and Finch, Robert **N6CXB** "A Duplex Packet Radio Repeater Approach to Layer One Efficiency," 7th **ARRL Amateur Radio Computer Networking Conference** pl [1988]
- [6] Chepponis, Mike **K3MC**, El-more, Glenn **N6GN**, Garbee, **Bdale N3EUA**, Karn, **Phil KA9Q**, Rowett, Kevin **N6RCE**, "The Implications of High-Speed RF Networking," 8th **ARRL Amateur Radio Computer Networking Conference** p 19 [1989]
- [7] **Elmore**, Glenn **N6GN**, Rowett, Kevin **N6RCE**, **Satterthwaite**, Ed **N6PLO** "Hubmaster: Cluster-Based Access to High-Speed Networks," 9th **ARRL Amateur Radio Computer Networking Conference** p 89 [1990]
- [8] Chepponis, Mike **K3MC**, Karn, Phil **KA9Q** "The KISS TNC: A simple Host-to-TNC communications protocol, " 6th **ARRL Amateur Radio Computer Networking Conference** p 38 [1987]
- [9] Heatherington, Dale A. **WA4DSY** "A 56 Kilobaud RF Modem," 6th **ARRL Amateur Radio Computer Networking Conference** p 68 [1987]

CHARACTERIZATION OF **PACSAT-1** TRAFFIC VIA **DOWNLINK** MONITORING

Robert J. Diersing, N5AHD
Department of Computer Information Systems
Texas A&I University

ABSTRACT

This paper provides a characterization of certain aspects of PACSAT-1 downlink traffic obtained by monitoring and processing the downlink data stream. Among the aspects presented are: the proportions of the three major traffic types--broadcasting, file server, and telemetry; the interarrival times of successful file server connections; the service times for file server transactions; the AX.25 data-link level response times for file server transactions; and the downlink byte counts for file server transactions. Even though the monitoring operation is subject to various factors affecting the received block and bit error rates, the data presented here should be good approximation of the parameters of the PACSAT-1 communications system.

INTRODUCTION

Between January, 1991 and July, 1991, a total of six downlink data samples were collected--four from PACSAT-1 and two from UoSAT-3. These are summarized in Table 1. This paper is concerned only with sample nos. 1(D), 2, and 3 which represent the data collected using a beam antenna (KLM-18C) and computer-controlled tracking. PACSAT-1 sample nos. 1(0) and 4 were collected using an omni-directional antenna (Cushcraft AFM-44DA). Details of the analysis of the PACSAT-1 omni-directional antenna samples and the UoSAT-3 samples will be published at a later date.

The "Percent of Capacity" column in Table 1 gives an estimate of the data captured assuming that the downlink was busy for the entire time the satellite was in view of the monitoring station. For PACSAT-1 the downlink is busy almost 100 percent of the time when its footprint is over the continental U.S. Even on an individual pass it is hard to account for more than about 90 percent of the downlink time. This is due to periodic pauses in downlink data transmission and bit errors in received frames caused by propagation anomalies and ground station local noise conditions.

DOWNLINK TRAFFIC MIX

Table 2 gives the frame and byte counts for each of the three primary traffic types--broadcast (QST), file server (BBS or FTL0), and telemetry (TLM). Digipeating constituted such a small part of the downlink traffic that it has been ignored in this analysis. The percentages are of the total byte count for the sample. Because the percentages in each category are relatively

Table 1
PACSAT-1 AND UoSAT-3 DOWNLINK DATA SAMPLE SUMMARY

Sample No.	Satellite Name	Time Period	Orbit Count	Downlink Time HH:MM:SS	Byte Count	Percent of Capacity
1(T)			52	09:49:59	3.6 Mb	68.1
1(O)	PACSAT-1	01/01 - 01/20	28	05:06:43	1.5 Mb	52.5
1(D)			24	04:43:16	2.1 Mb	04.9
2	PACSAT-1	02/01 - 02/28	48	09:40:47	4.4 Mb	83.4
3	PACSAT-1	03/17 - 05/29	32	06:41:28	3.2 Mb	87.7
4	PACSAT-1	03/23 - 06/14	32	06:37:14	1.5 Mb	42.7
5	UoSAT-3	03/04 - 04/29	27	05:19:06	12.7 Mb	55.2
6	UoSAT-3	05/05 - 07/14	29	06:05:41	7.6 Mb	28.9

uniform, it would be interesting to further subdivide the traffic into a few more categories. For example, daytime versus evening and weekday versus weekend.

Table 2
PACSAT-1 DOWNLINK TRAFFIC MIX

	Sample No. 1(D)	Sample No. 2	Sample No. 3
QST			
Frames	6,272	12,081	11,860
Bytes	981,891	1,915,835	1,630,643
Percent	45	44	51
BBS			
Frames	9,706	19,226	10,281
Bytes	1,062,230	2,150,138	1,353,752
Percent	49	49	43
TLM			
Frames	1,409	3,261	2,211
Bytes	120,108	293,132	184,259
Percent	6	7	6
TOTAL			
Frames	17,387	34,568	24,352
Bytes	2,164,229	4,359,105	3,168,924

FILE SERVER CONNECTION INTERARRIVAL TIMES AND SERVICE TIMES

File server connection interarrival time and service time statistics are given in this section. For these and the other measurements described later, the sample mean and standard deviation were computed and an empirical cumulative density function (CDF) was plotted. Then, depending on the values of the mean and standard deviation and the shape of the CDF, a Chi-Square test for goodness of fit of the data with a hypothesized distribution was made. Usually the sample data was tested for

either an exponential or normal distribution with parameters estimated from the sample being tested.

For the data presented here, all of the Chi-Square tests resulted in rejection of H_0 , that the sample data fit the distribution of interest. However, for some of the cases, the resulting Chi-Square statistic was very near the critical value. Consequently, the Chi-Square statistic closest to the critical value is reported in each case along with the type of test being performed. If "E" appears in the " χ^2 Stat" column, the value given resulted from a test for an exponential distribution. An "N" means that the test was for a normal distribution. For easy reference, the following critical values are included: $\chi^2_{0.01,8} = 20.1$ and $\chi^2_{0.05,8} = 15.5$ for the exponential test and $\chi^2_{0.01,7} = 18.5$ and $\chi^2_{0.05,7} = 14.1$ for the normal test. Other values may be found in any text on statistics or simulation.

Successful Connect Request Interarrival Times

The time between successful file server connections was determined by measuring the time between state changes in the BBSTAT message. These measurements are subject to any time delays occurring between the actual AX.25 connection and the time of appearance of the BBSTAT message on the downlink. Moreover, the monitoring station could miss a BBSTAT frame. Table 3 gives a summary of the means and standard deviations for successful connection interarrival times. Figure 1 shows the exponential distribution of interarrival times from sample no. 3.

Table 3
SUMMARY OF SUCCESSFUL CONNECTION
INTERARRIVAL TIME ANALYSIS

Sample No.	Number of Observations	Mean (Seconds)	Std Dev (Seconds)	χ^2 Stat and Test
1 (D)	299	61	45	53.1 E
2	682	54	45	81.7 E
3	477	55	49	44.2 E

File Server Connection Service Times

When the downlink AX.25 frame sequence indicated the start of a new file server connection, the time was noted. After that, the arrival time of each frame was logged. When the frame sequence indicated that the connection had terminated, the starting time of the connection was subtracted from the last frame time giving the service time for the transaction. In cases where LOS occurred before the connection terminated, the service time was computed based on the time of the last received frame.

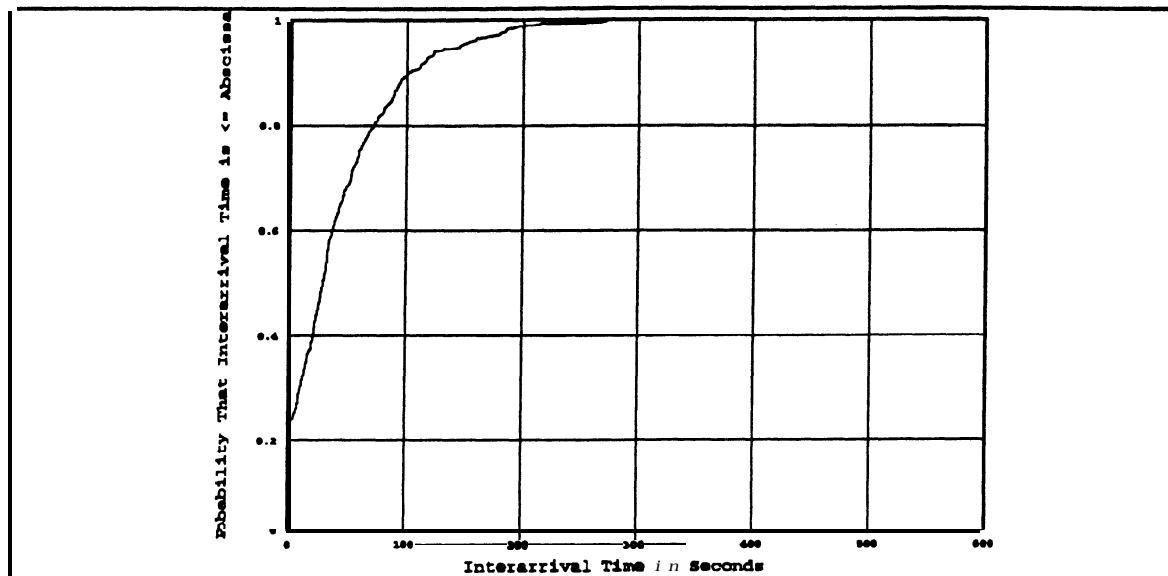


Figure 1. Empirical CDF for FTLO successful connect interarrival times constructed from PACSAT-1 data sample no. 3. $\bar{y} = 55$, $s = 49$.

However, only connections that started while the satellite was in view of the monitoring station were counted.

A summary of the transaction service time statistics can be found in Table 4. A representative empirical CDF for sample no. 3 is shown in Figure 2.

Table 4
SUMMARY OF FTLO TRANSACTION SERVICE TIME ANALYSIS

Sample No.	Number of Observations	Mean	Standard Deviation	χ^2 Stat and Test
1(D)	228	128	104	22.8 E
2	472	115	90	42.6 E
3	330	108	95	35.2 E

AX.25 DATA-LINK LEVEL STATISTICS FOR FILE SERVER CONNECTIONS

Two data-link level statistics were computed for file server connections--the response time and the byte count. For an individual connection, the response time is the average time between AX.25 frames associated with the client/server connection. Note that this does not necessarily correspond to what the user may perceive as response time since there may be more than one data-link level exchange before there is any noticeable "response" observed by the ground station operator. This is particularly true when error-recovery procedures have been executed at the data-link level.

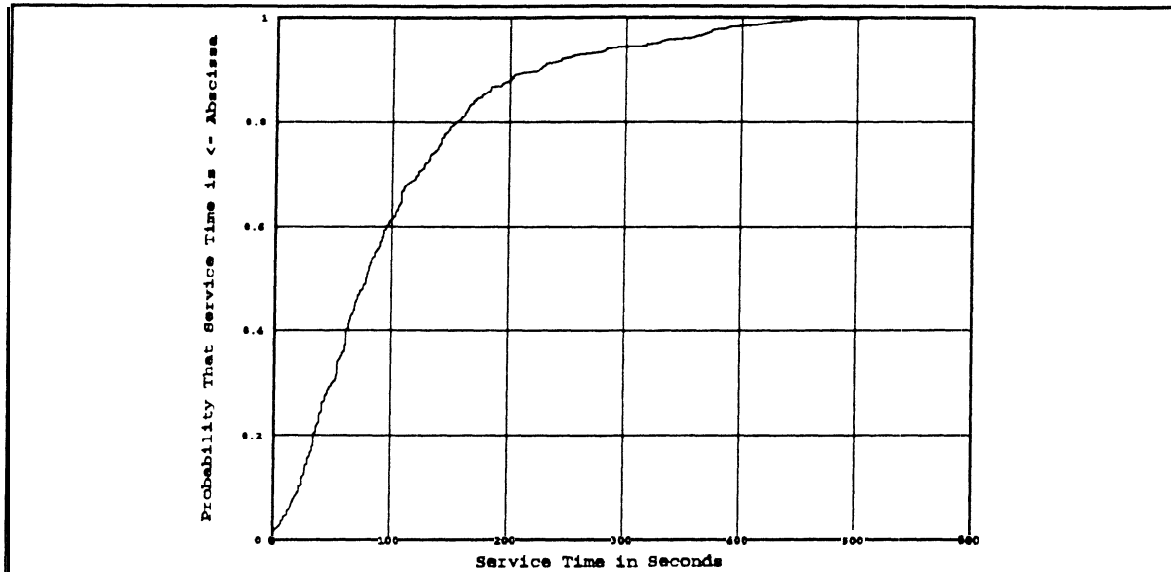


Figure 2. Empirical CDF for FTLO transaction service times constructed from PACSAT-1 data sample no. 3. $\bar{y} = 108$, $s = 95$.

AX.25 Response Time Per Connection

The AX.25 response time per connection is summarized in Table 5. An empirical CDF constructed by combining all three PACSAT-1 data samples appears in Figure 3. The CDF has the general shape of a CDF for a normal distribution. The Ax.25 response time data was the only data exhibiting a normal distribution. The mean value for response time given in Table 5 is the average of the individual connection averages. Recall that while any given connection is being measured, the file server could be servicing as many as three other clients and telemetry and broadcast frames are also being generated on the downlink. A better response time measurement could be obtained if both uplink and downlink traffic could be monitored at the same time. However, due to the transaction-oriented operation of the client/server system, there is no operator keyboard response time to be considered.

Table 5
Ax.25 RESPONSE TIMES FOR FTLO CONNECTIONS
MEASURED ON A PER-CONNECTION BASIS

Sample No.	Number of Observations	Mean	Standard Deviation	χ^2 Stat and Test
1 (D)	225	5.0	2.3	31.0 N
2	464	5.1	2.6	42.3 N
3	326	5.2	3.0	54.3 N

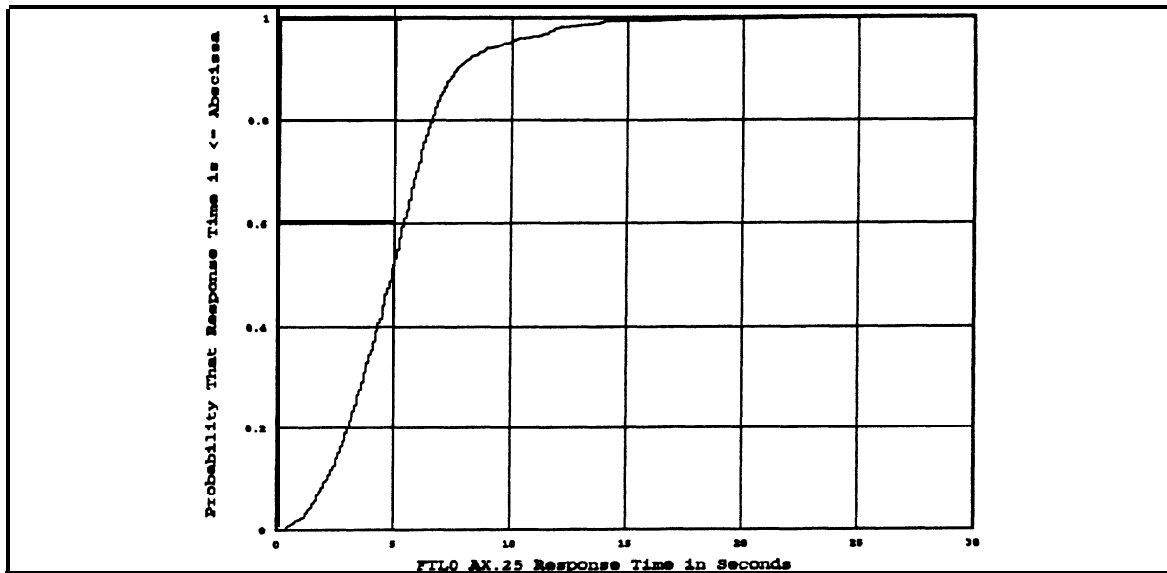


Figure 3. Empirical CDF for FTL0 AX.25 response times constructed by combining all PACSAT-1 data samples.

AX.25 Frame Byte Counts Per Connection

Table 6 contains a summary of the AX.25 frame byte counts for file server connections. The empirical CDF for the distribution of byte counts for all PACSAT-1 samples combined is shown in Figure 4. Notice that the goodness-of-fit tests of the byte count data resulted in **Chi-Square** statistics closer to the critical values when compared to the results of the other tests. Data frames for file server connections seen on the **downlink** are primarily responses to directory and download commands issued by the client stations. Only connections where at least 128 bytes were sent to the client station were counted in these measurements.

Table 6 BYTE COUNTS FOR FTL0 OPERATIONS MEASURED ON A PER-CONNECTION BASIS				
Sample No.	Number of Observations	Mean	Standard Deviation	χ^2 Stat and Test
1 (D)	212	3,120	3,371	16.1 E
2	446	2,637	2,845	34.8 E
3	306	2,700	3,692	20.6 E

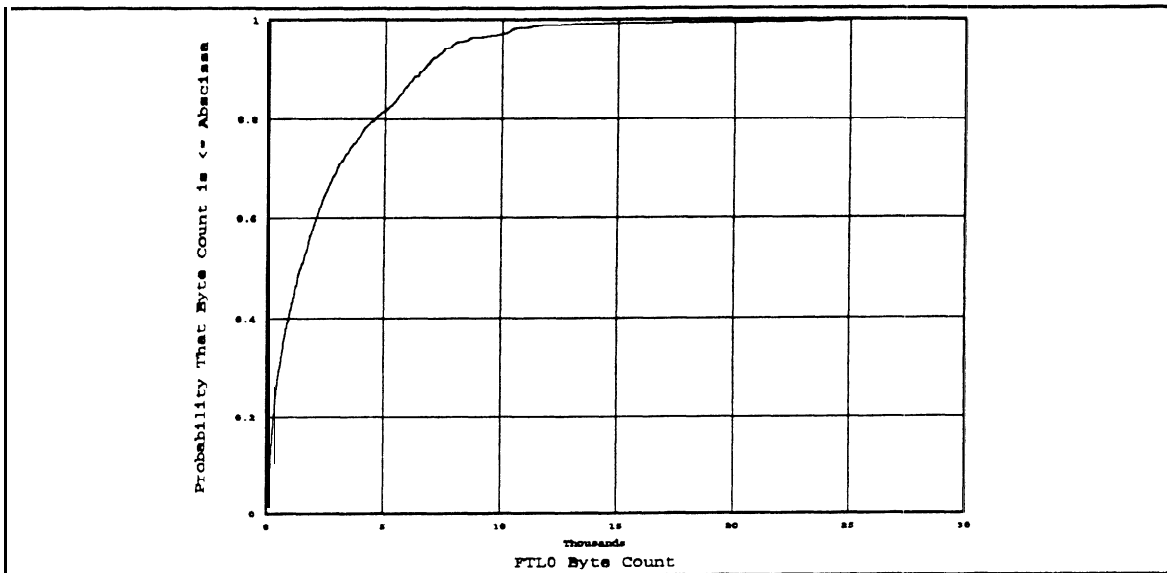


Figure 4. Empirical CDF for FTLO transaction byte counts constructed by combining all PACSAT-1 data samples.

SUMMARY

The statistics reported in this paper have been compiled in preparation for construction of a simulation model of communication systems similar to those on PACSAT-1 and UoSAT-3. The information provides an operational characterization from the viewpoint of a ground station user. Even though the monitoring process is subject to missing data due to bit errors in received frames, the data samples are large enough and span a long enough time period that they should fairly represent actual operating conditions. In order to minimize the number of errors encountered, only data captured using computer-pointed directional antenna systems have been used during the compilation of the preceding statistics.

EXPERIMENTAL STUDY OF SHANNON-FANO, HUFFMAN, LEMPEL-ZIV-WELCH AND OTHER LOSSLESS ALGORITHMS

D. Dueck and W. Kinsner, VE4WK

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, **Manitoba**, Canada **R3T-2N2**
Fax: (204) **275-0261**
e-Mail: **Kinsner@ccm.UManitoba.CA**
E-mail: **VE4WK@VE4KV.MB.CAN.NA**

Abstract

Knowledge of the statistics of a source bit stream is required when selecting the most efficient statistical data compression techniques and designing the best codes. This paper presents a program called Statistical Analysis of Files (**STAF**) that analyzes and supplies the statistics on such bit streams. The two key statistics are the **entropy** (a measure of information content) and the **frequency of occurrence** table. The entropy measure is used in establishing the compression limit for statistical techniques, while the frequency of occurrence is vital in the designing of optimal variable-length codes such as **Huffman** and **Shannon-Fano**. Other statistics analyzed for optimal code compression techniques are run-length encoding, half-byte packing, and **diatomic** character encoding. In the entropy report, the statistical techniques are compared with a popular adaptive nonstatistical dictionary encoding algorithm, the **LZW** technique, to give a comparison with other methods of **lossless** compression on a set of benchmark files.

1. INTRODUCTION

Transmission of information requires symbols (data). If the data representation of information is compact (i.e., no redundancy is present) the information can be transferred faster than with redundant data, given the same data transfer rate, expressed in bits per second (bps). Thus, every bit of information is **transferrable** to others, at the right price. A Stocks and Bonds newspaper can be very expensive to send, but the price is worth it -to a stockbroker. Similarly, last night's sports scores are very important to the sports fan, but for someone who dislikes sports, the information is useless.

Information can be sent by a variety of methods, and each method has its own benefits and pitfalls. For a given probability of error, a shorter transmission would be less susceptible to error than its longer corresponding uncrunched source. Nevertheless, the shorter bit stream is more sensitive to **errors**, such that an error of one bit could destroy the whole information, whereas an error in the uncompressed bit stream could likely be corrected. Which then, is the best method? Clearly, the best **method** is the fastest, most perfect method, though which one is best for a particular type of **information** is the **\$64M** question. Data compression can be employed to reduce the **Length** of the data bit stream, without losing any of the **information** itself. Most data is actually very repetitive (redundant), and can be "crunched" so **as** to remove the repetitiveness. A good elementary example of data compression is secretarial short-hand. People who know the code can transfer information quickly **from** a spoken format into a written format. Then at a more convenient time, the code can be expanded into a normal textual format. Elimination of repetitiveness, or redundancy, then is an improvement to **information** transmission.

Information exchange fills each of our days. If it can be improved in a manner that does not hinder our understanding, it should be done. Information can be represented in a variety of ways, such as pictures, speech, books, television, and computer files. All of these **formats**, (and many others) require a transfer **from** a source, to a destination. Thus, the transfer is what can be improved, and just about everyone can appropriate it.

In selecting or designing proper statistical data compression methods and techniques, the knowledge of the statistical properties of the data is essential. A thorough review of exact and lossy data compression methods and techniques was recently presented by Kinsner [Kins91a], [Kins91b], [Kins91c]. The exact techniques compress and reconstruct source bit stream without any losses, and are required for transfer of critical data such as programs and financial files. The best techniques in this category require knowledge about the statistical properties of the bit stream to be compressed. The key statistical property is the frequency of occurrence of each pattern in the stream.

A program designed to analyze any bit stream and provide the necessary **statistical** information for code design has recently been described [DuKi91]. The Statistical Analysis of Files (STAF) program provides the user with a portrait of a file's statistics. Then, on the basis of the report, the user can then decide which type of data compression **would** be the most suitable for their particular piece of data. The STAF program generates two types of reports: a short and a full-length report. The short report is a one or two page report containing the following two segments: (i) the entropy analysis, and (ii) the sorted frequency of occurrence. The full report gives the following results: (i) entropy analysis, (ii) a full character frequency report, (iii) the half-byte, (iv) run-length encoding, and (v) diatomic character analyses.

3. DESCRIPTION OF PROGRAM MODULES

The STAF computer program has five modules. The two types of reports (short and full) are similar in their scope: The short report gives the entropy analysis and the sorted character frequency chart, while the long report gives additional half-byte analysis, **repeated** character string analysis, diatomic analysis, and the critical ASCII frequency of occurrence; chart.

3.1 Entropy Report Module

The entropy analysis module is divided into an uncompressed and compressed section. The **uncompressed analysis** is the most basic part of the program. Firstly,, it gives the count of the total number of characters in the file, which is also converted into bits. Next the number of distinct characters in the file is printed. Finally, the source, entropy, **H_s** , is calculated [Kins91c]. The **compressed** analysis gives six important statistics useful for determining whether a statistical compression method is feasible. The statistics **are** described next.

Theoretical Statistical Compression

Let us consider a source containing 200 characters: 100 **Es**, 50 each of T and A. The entropy is calculated as

$$\begin{aligned}
 H_s &= -\sum_{i=1}^m p_i \log_2 p_i \\
 &= -(0.50 \log_2 0.50 + 2 \times 0.25 \log_2 0.25) \\
 &= 1.5 \text{ bits/character}
 \end{aligned}
 \tag{3.1}$$

where p_i is the probability of occurrence of each symbol in the source, and m is the number of symbols in the source text or alphabet (hereafter referred to as **source**).

The ultimate theoretical statistical compression (**TSC**) is given as the barrier which every statistical code seeks to equal, but is rarely, if ever achieved. The ultimate **TSC** percentage, U_p , is **calculated from**

$$\begin{aligned} U_p &= \frac{S_a - H_a}{S_a} \times 100 \\ &= \frac{8 - 1.5}{8} \\ &= 81.25 \% \end{aligned} \quad (3.2)$$

where S_a is the number of bits used to represent a character (normally eight, according to ASCII convention), and H_a is the source code entropy. Notice for this example we have a perfect code, one that is statistically as concise as possible. But in normal practice, a three symbol code is next to useless, and a larger and undoubtedly less perfect code would be created with a larger symbol set.

Theoretical Variable Length Codewords

While the theoretical statistical compression, U , is the standard to measure up to, the theoretical variable-length codes entropy (V) is the estimate of the entropy when variable length coding is used. While it would be necessary to actually construct the Shannon-Fano or **Huffman** codes to actually know the number of bits needed to encode each symbol, the **estimated** number of bits, λ , required to encode a symbol with probability Of p_i is

$$\lambda_i = \lceil \log_2 p_i \rceil \quad (3.3)$$

The theoretical variable length **code** entropy, V_H , **can be** calculated **from**

$$V_H = - \sum_{i=1}^m p_i \lambda_i \quad (3.4)$$

In practice, the actual S-F or **Huffman** code entropy would approach, but never be smaller than U_H , such that

$$U_H \leq S\text{-}F_{Hc} \text{ or } \text{Huff}_{Hc} \leq V_H \quad (2.8)$$

is always true.

Tk Shannon-Fano Compression Technique

The Shannon-Fano (S-F) coding module calculates a possible S-F code and the code entropy. A separate program was developed to calculate a number of S-F codes using a number of different heuristics, but one heuristic consistently created the best code every time, so the STAF program uses only this heuristic. Once the entropy is calculated, it is a simple matter to calculate the length of a file **coded** in this manner, and the percentage compression that may be gained with this method

The heuristic used to calculate the S-F tree is this: Starting **from** the top down, the whole array containing the sorted character frequency from 0 to the number of distinct characters (symbols) is scanned, and split in half as close as possible to the middle of the frequency. The program splits the top half, (with fewer characters, but appearing more often) and **gives** each character in this section a prefix of “zero”, and the bottom half a “one.” **Now** the array is split into two approximately equivalent parts (**according** to frequency), and the same heuristic for splitting the initial **array** in half is called again, recursively, to split the top, and bottom halves of the array, and add the next character in the S-F code, until all the codes have been generated.

In the routine that actually finds the middle of the **array**, a check is added that has been coined “Heat-Seeking; Capability”. Just as a heat-seeking missile or torpedo searches for a source of heat as its target, so this “Find the middle of the array” routine seeks the: closest point to the middle. The less advanced “Find the Middle” heuristics checks the array, returning the point exactly in the middle or earlier, while the heat seeking capability checks on both sides of the exact middle point of the array, and returns the closest point, whether it **be** on the top, or bottom side of the array. The old method can be compared to the way contestants had to guess the prices of items on the popular T.V. game show “The Price Is Right”, where the winning answer was “The contestant closest to the actual retail price, but not over is...”. This method seems a little unfair because the winner was not necessarily the one who was the closest, but the closest one who guessed underneath the actual price. The “Heat-Seeking” capability eliminates the disparity, and returns the split **closest to the middle**.

The Shannon-Fano codes are pleasing to display and analyze, since it is very apparent that the code is self-separating. **Huffman** codes (explained next) are more complicated in creation, and this results in a code that is not as obviously self separating.

The Huffman Compression Technique

The **Huffman** codes are created using a single heuristic. Just as in the S-F module, a separate program was developed which coded a number of **Huffman** codes and compared them. For each file tested, the same heuristic gave the best result. Each code had the same entropy, but one heuristic produced codes where the maximum code length was less than others. This is the heuristic that has been implemented in the program. The **Huffman** code entropy is calculated, and this appears in the entropy report module.

Briefly, **Huffman** codes are created by fusion [Huff52], [Kins91a]. Shannon-Fano codes are created by splitting an array into successive halves, quarters, (etc... (called “top-down splitting”), while **Huffman** codes are created by repeatedly (recursively) merging the two symbols with the smallest frequency, creating a new entry with the combined **frequency** (called “bottom-up binary fusing”) and adding this new node to the list/array from which the two symbols were located. The two individual symbols are also removed from the list, since they are now represented by a single combined (and thus higher) frequency.

This process **continues**, always subtracting two nodes from the list, and merging their composite probability back to the list, until one large binary tree is formed. The **Huffman** code can then be read from the: tree, starting from its corresponding leaf going up through the branches to the root, where each step up “left” is assigned a “one”, and each step up “right” is assigned a “zero”. A more detailed description, with examples of both **Huffman** and S-F codes is given in [Kins91a].

The LZW Compression Technique

This algorithm is a non-statistical dictionary algorithm, and thus it is possible (and **likely**)

that coding in this case could exceed the symbol based entropy, H_a . **Currently the** only available statistics are those compiled by actually running the source bit stream **through** this algorithm. The technique creates a dictionary, and this heuristic **clears** the dictionary when it fills up. Since the figures included in the entropy module are for comparison against the statistical techniques, a detailed description of the **algorithm** can be found in [KiGr91], [Kins91b], [Welc84], [Stor88].

3.2 Frequency Report Module

This second module in the STAF program provides the user with two different types of charts: (i) standard, and (ii) sorted charts. The full length report calls for both charts, while the short report calls for only the **sorted chart**.

Sorted Frequency Chart

The sorted frequency chart looks at the whole **file** to be analyzed, and sorts the symbols appearing in the source file in a descending order. Characters in the normal ASCII character set which do not appear in the file are not shown, for clarity. The chart shows each character, the integer count of the number of times it appears in the source, and the frequency or percentage, p_i , of the file that is that specific character. A typical text file's sorted chart would probably begin with <Space>, followed probably by <CR> (carriage return) and then perhaps the frequent letters **E, T, A**, and so on. Recall that the Shannon-Fano and **Huffman** codes are constructed on the basis of this chart.

Standard Frequency Chart

The Standard Frequency Chart prints the entire **256-character** ASCII character set each with its corresponding count and frequency. This chart would be useful to show how the different characters are used, and which, if any blocks of characters are either used extensively or not at all. A typical Analog-to-Digital converter supplying **8-bit** data could create data where this chart or a graphical representation of it could be useful in conjunction with another type of data compression using patterns derived from inspection of the chart. Coding could take advantage of "clusters" of certain symbols being used more often, and codes could be calculated accordingly.

3.3 Half-Byte Encoding Module

Certain types of data contain extensive numerical figures. Business charts, spreadsheets, **scientific** measurements (or even the sorted frequency charts the STAF **program** generates) contain many numerical figures which could be compressed. Normally, a character byte takes eight-bits (S_a), but if the compression algorithm anticipates a numerical string, a four-bit code could be **utilized to encode the digits zero to nine, creating a compression ratio** of nearly **50%**.

This four-bit code means 16 characters can be encoded this way, so that after the ten digits are assigned, there are six extra unused codes, which could be classified as "numerical" and encoded as such. For example, a phone number (**1-800-555-1212**) contains 14 characters, including the three hyphens. This could be encoded into nine characters (A starting control byte, length of string, and the 14 nibbles) for a 64% saving. For longer numerical strings, a **savings** approaching **50%** could **be** achieved. Typically a compression program would assign the six extra characters to be ones that normally show up in association with numbers. ("**\$*/-.,**"). Thus, the following types of strings would be encodable: 1-800-668-1234, **1984,1988,1990-91, 08/07/91-5/01/68, \$***** 1,234.56 and 3-2212-01176-3284. For highly statistical and numerical files, a **compression** of up to **50%** could be attained with half-byte encoding.

3.4 Run-Length Encoding Module

Run-length encoding (RLE) is a very simple technique, useful for highly repetitive characters strings. The first type of RLE is when the encoder anticipates a number of blanks. These can **be replaced** with two characters: a special character, and the count, n, of the number of blanks. Thus, in a chart or graph that has 10 blanks in a row, they could be encrypted as **a special control signal character**, and then the number **ten**, to signify “10 blanks in a row.” An extrapolation of this technique is a three character **code** used to represent repeated characters. For example, if the encoder encounters six Rs, they could be encrypted as a [special **character**| the **original character**|repeated 6 times].

Normally it is **quite** rare to have more than two or three characters repeated in a row, so this technique would not be a very fruitful one. Multiple spaces however, are a little more common, and are prime candidates for compression. This module only counts repeated strings of three or more characters, **because** compression savings only start with repetitions of more than **three** characters.

3.5 Diatomic Analysis Module

The **diatomic** compression technique is another very specific technique, which takes certain strings of characters and replaces them with a shorter code. In this case, the program looks for all **pairs** of characters, and replaces the most common pairs with a single character code. This results in a compression percentage of **50%**, though it probably would not be possible because of the limited number of available characters to signify pairs of characters. Thus only the most common pairs would be encrypted. The most common pairs in the english language are E_, _T, TH, _A, and S_ [Held87].

4. EXPERIMENTAL RESULTS

Experiments were performed on a variety of files, most for the IBM PC. The files were chosen for benchmarking the results. The STAF program was run on a **33-MHz** IBM Compatible computer, and output sent to an ASCII **file** on disk. Table 4.1 summarizes the output the STAF program created on each of the files.

4.1 Benchmark Files Description

README.DOC - This text file contains mostly upper and lowercase letters, with a sprinkling of a few numbers. It is Borland's Turbo C++ help file containing the last minute changes to their product. The **file** contains 82 distinct characters, and would represent a typical word processor letter or document.

AMHELP2.HLP - The Windows 3.0 HELP reference file for the Desktop Publishing program **AMI**, available to the PC user when <F1> is pressed. The file consists of mostly ASCII text, but contains a generous sprinkling of control and non-printable **characters.necessary** for Windows to be **able** to interpret it,

WORKS.EXE - One of two executable files tested. This is **the .EXE code from** the popular **Word-Processing/DataBase/Spreadsheet/Communications** software package from Microsoft. This fits in the category of non-windows applications.

MILLEBV4.EXE - Another executable file, this one is an implementation of the popular card game Mille **Bornes**. A good mouse-based, windowed (Not MS 'Windows) program with excellent graphics.

Table 4.1 STAF analysis of Benchmark Files

FILE	Entropy / Compr Len / Compr Pct	Theoretical Var. Length Codes	Shannon-Fano Coding	Huffman Coding	LZW Coding
README.DOC (16203 bytes) (82 symbols)	4.825 bits/char 9773 bytes 39.7 %	5.362 10861 32.97 9b	4.92168 9968 38.479 %	4.8565 1 9836 39.294%	3.904 7908 51.194 %
AMIHLP2.HLP (279655 bytes) (251 symbols)	4.898 bits/char 171232 bytes 38.77%	5.337 186561 33.289 %	4.94101 172722 38.237%	4.93135 172384 38.358%	2.870 100329 64.124 %
WORKS.EXE (381776 bytes) (256 symbols)	7.409 bits/char 353556 bytes 7.392%	7.956 379676 0.55%	7.45306 355674 6.837%	7.43783 354947 7.027 %	7.822 373275 2.227%
MILLEBV4.EXE (216455 bytes) (256 symbols)	7.142 bits/char 193254 bytes 10.719%	7.697 208250 3.791%	7.23544 195768 9.557 %	7.16612 193892 10.424 %	5.934 160549 25.828%
TIGER.BMP (212086 bytes) (55 symbols)	1.065 bits/char 28246 bytes 86.682%	1.292 34246 bytes 83.853 %	1.23237 32671 bytes 84.595 %	1.23224 32667 bytes 84.597 %	0.153 4069 bytes 98.081%
FIT.PCM (20556 bytes) (202 symbols)	6.469 bits/char 16622 bytes 19.14%	17997 7.004 bytes 12.45 %	6.587 16924 bytes 17.67 %	6.4998 16701 bytes 18.75 %	6.279 16134 bytes 21.51 %
NUMBERS.TXT (16738 bytes) (226 chars)	3.733 bits/char 7810 bytes 53.342 %	4.375 9155 bytes 45.307 %	3.78104 7910 bytes 52.737 %	3.75881 7864 bytes 53.015%	2.534 5302 bytes 68.324 %
SHORT.TXT (894 bytes) (34 symbols)	4.137 bits/char 463 bytes 48.294%	4.582 512 bytes 42.729 %	4.1868 467 bytes 47.679 %	4.1656 465 bytes 47.945 %	5.038 563 bytes 37.025%

v5

TIGER.BMP - A very simple Windows 3.0 Paintbrush drawing of a cat's face. Although this may not be as detailed as a typical drawing, it illustrates the waste that can be eliminated from every drawing.

FIT.PCM - A digital voice sample of a person speaking the word "FIT." The sample is stored in pulse code modulation (PCM) form. This sample bit stream would appear random, unless displayed in graphical PCM format.

NUMBERS.TXT - The STAF program creates reports which contain many numbers. The file NUMBERS.TXT is actually the STAF report for the README.DOC file. The sample of this

file can be created by running STAF on **README.DOC**, and then running **STAF** on the output file **README.ST2**, after **first** renaming it to a **different** name.

SHORT.TXT - A shorter memo, created to illustrate **Huffman** and S-F codes. It contains roughly 30 distinct symbols, and is used to illustrate these two types of coding simply.

4.2 Observations

The specific implementation of the **Huffman** code is better than the: S-F code in every instance of the benchmarks that were tested. From the selected **files**, it appears that the more symbols in **the** source bit stream, the smaller the difference becomes. The two EXE **file** (**WORKS**, **MILLEBV4**), each using all 256 **8-bit** codes exhibit differences in **code** entropy of 0.015 and 0.07 bits/character. The two text files on the other hand (**README.DOC** (82 symbols) and **NUMBERS.TXT** (226 symbols) exhibit differences in code entropy of **0.65** and 0.02. The difference decreases in proportion to the number of symbols in the bit stream. The **AMIHELP2.HLP** file also supports this observation: It too has more than 250 symbols in the file, and the **Huffman** and S-F difference in entropy is approximately 0.01.

The **TIGER.BMP** picture is quite simple, and comes in with an entropy of a resounding 1.065 bits/character. The **Huffman** and S-F code entropy is virtually identical, though the S-F entropy is still larger.

The **file SHORT.TXT** is a small paragraph typed in from a book, using lowercase letters only. Containing 30 characters, it is a short memo, illustrating clearly **Huffman** and S-F codes that can easily be reconstructed. The **LZW (37.%)** algorithm does not appear to perform as efficiently as the variable length coding routines (48%). This is due to the brevity of the file. If the **file** becomes any larger, the **LZW** will perform better, since it will be able to find more redundancy and patterns in the text.

The **LZW** compression algorithm is non-statistical, and thus its statistics can really only be used for comparison. The **LZW** algorithm is better for each file by approximately **15-20%**, except for the **WORKS.EXE** program, where the LZW routine makes a remarkably poor showing. Typical LZW gives compressions of **50-60%** for normal text **files**, and only **20-30%** for more evenly (or random) distributed probabilities like the **.EXE** files and digitized speech. The LZW routine compresses and amazing 98% on the **TIGER.BMP** picture, though the picture is a relatively simple, **8-color** diagram. Nevertheless, all BMP stored format pictures can be compressed - some more than others.

5. CONCLUSIONS

This paper presents a statistical analysis of files computer program developed to design optimal statistical codes for data compression. A set of benchmark files has been selected to test relative merits of the **Shannon-Fano** (S-F), **Huffman**, and Lempel-Ziv-Welch (**LZW**) optimal codes. Entropy calculations and frequency of occurrence help in assessing the best statistical techniques that can be applied to the given data stream. The frequency of occurrence is used to design optimal **Huffman** and S-F codes for that stream. The optimal codes are designed on a set of heuristics also evaluated in the study. The specific implementation of the **Huffman** code appears to **better** than the optimal S-F code for all the **files** tested. Our implementation of the LZW algorithm is also better than **Huffman** by **15-20%**. The LZW gives compressions of **50-60%** for normal text **files**, and **20-30%** for executable files. This statistical analysis of files program could be used as a tool in designing and implementing compression algorithms in future packet radio **BBSs** and other data transfer systems.

ACKNOWLEDGEMENTS

This **work** was supported in part by the University of Manitoba and the Natural Sciences and Engineering Research Council (**NSERC**) of Canada.

REFERENCES

- [DuKi91] D. Dueck and W. Kinsner, "A program for **statistical** analysis of **files**," *Technical Report, DEL91-8*, Aug. 1991, 50 pp.
- [Held87] G. Held, *Data Compression: Techniques and Applications, Hardware and Software Considerations*. New York (NY): Wiley, 1987 (2nd ed.), 206 pp. (QA76.9.D33H44 1987)
- [Huff52] D.A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, pp. 1098-1101, Sept. 1952.
- [KiGr91] W. Kinsner and R.H. Greenfield, "The Lempel-Ziv-Welch (**LZW**) data compression algorithm for packet radio," *Proc. IEEE Conf. Computer, Power, and Communications Systems*, (Regina, SK; May. 29-30, 1991), 225-229 pp., 1991.
- [Kins91a] W. Kinsner, "Review of data compression methods, including Shannon-Fano, Huffman, arithmetic, Storer, Lempel-Ziv-Welch, **fractal**, neural network, and **wavelet algorithms**," *Technical Report, DEL91-1*, Jan. 1991, 157 pp.
- [Kins91b] W. Kinsner, "Lossless and lossy data compression including **fractals** and neural networks," *Proc. Int. Conf. Computer, Electronics, Communication, Control*, (Calgary, AB; Apr. 8-10, 1991), 130-137 pp., 1991.
- [Kins91c] W. Kinsner, "Lossless data compression for packet **radio**," *Proc. 10th Computer Networking Conf.*, (San Jose, CA; Sept. 29-30, 1991), this Proceedings, 1991.
- [Stor88] J.A. Storer, *Data Compression: Method and Theory*. New York (NY): Computer Science Press/W.H. Freeman, 1988, 413 pp. (QA76.9.D33S76 1988)
- [Welc84] T.A. Welch, "A technique for high-performance data compression," *IEEE Computer*, vol. 17, pp. 8-19, June 1984.

Recent Hubmaster Networking Progress in Northern California

Glenn Elmore N6GN
Kevin Rowett N6RCE

Abstract

This paper describes the authors' progress toward implementing a prototype digital amateur network in Northern California. A goal of the network is to provide network layer user access and high enough performance to allow a wide range of applications, both old and new, to coexist on a single network built and maintained by all users.

Papers in previous proceedings from this conference have addressed the implications,^{[1][2]} portions of an implementation,^{[3][4]} and a proposed protocol^[4] of a higher speed wide area amateur radio network.

This paper is a brief description of some of our further progress towards implementing a prototype network in Northern California and of some of the practical issues which must be addressed when a wide area and medium performance amateur network is built.

“Coordination” is truly the key word with regard to building a significant and effective amateur radio network.

Our goal has been to prototype a network providing Layer 3 access by users and providing moderate performance, speed and latency for

demonstration of concept and use by Northern California amateurs.

As we have written before, the task of coordinating a physically distributed and diverse group of resources is proving to be the most difficult aspect of developing a prototype network. As interesting and challenging as they are, the physical fundamentals, hardware design and communication protocols needed to build a network are only a part of the overall problem. Finding ways to coordinate at all levels and thereby optimally apply available resources is proving to be the common theme.

“Coordination” is truly the key word with regard to building a significant and effective amateur radio network. For this reason, examining the necessary aspects of coordination at various layers of the OSI networking model is useful for describing our progress locally as well as describing some of what others will need to do if the dream of a wide area moderate performance digital amateur

network is to become a reality.

Physical Layer

Directional Antennas and Installation

A previous paper^[2] has described why directive antennas and higher frequencies must be used to achieve our goals. Directional antennas are inherently a form of coordination since they must be pointed in the direction of the other end(s) of a communication link.

Since our last report we have put considerable effort into building and measuring radios and antennas for the 33 cm band. We have also made measurement of path loss over both line-of-sight (LOS) paths and paths obstructed by trees and other vegetation. The effect of obstruction on distortion of digital data streams has been observed.

Several of the 13 element yagi antennas previously described have been independently built by N6GN, N6RCE and N6OLD. The first design resulted in about the correct maximum gain, 15.3 dBi, but at approximately 2% lower than the desired frequency. The next version was modified to correct this and vector network analyzer measurement of an identical pair as well as remote measurement with a 10W transmitter, indicated greater than 15 dBi gain for each antenna.

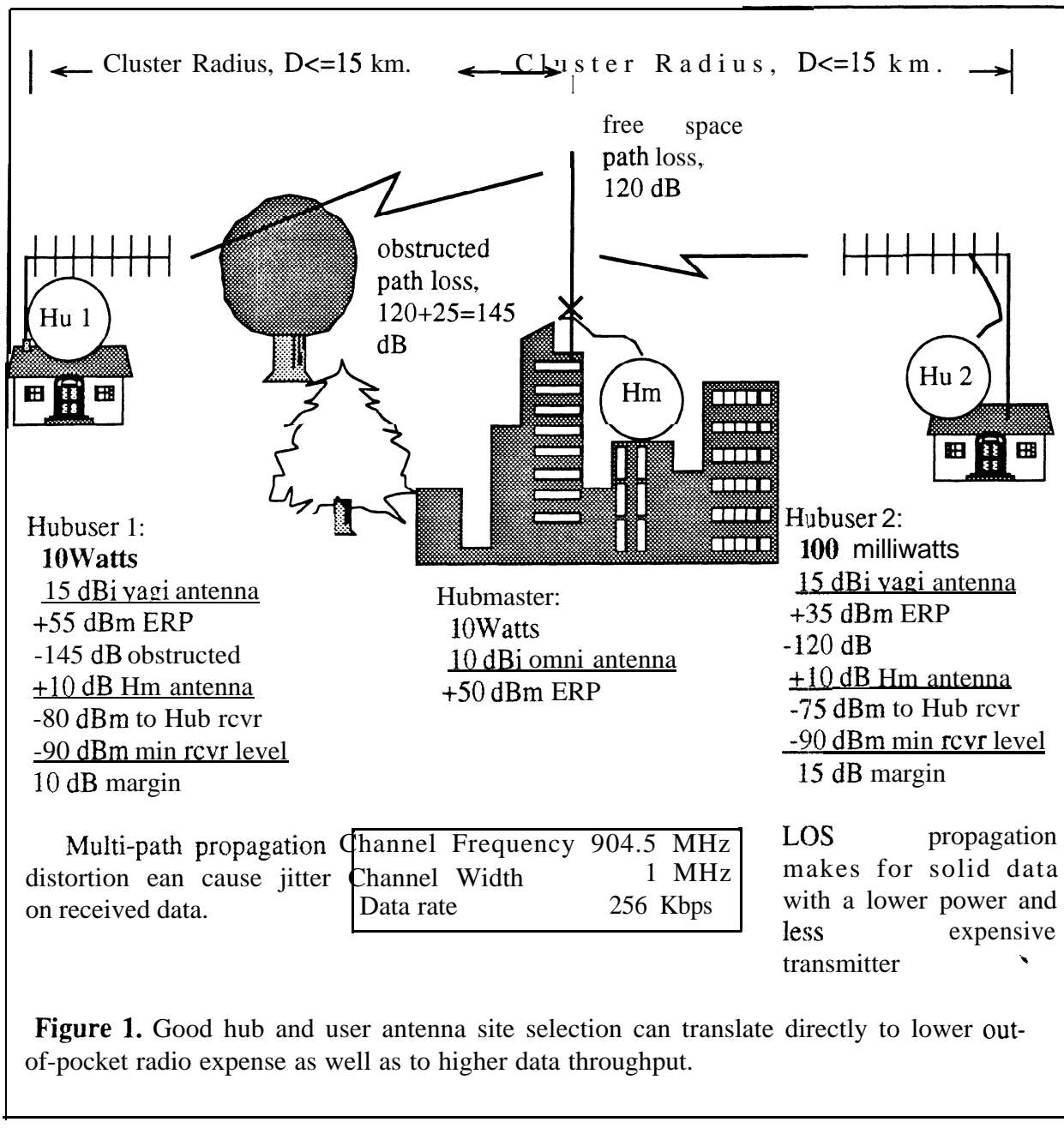
A commercial colinear antenna was purchased for use at the center of one of the Hubmaster clusters. It was also measured and showed a gain of approximately 11 dBi which, while considerably less than the manufacturer's published specification, is about correct for its size as anticipated.

Measurements of our 10 watt transmitters over LOS paths, from 50 feet to 70 miles, has basically confirmed antenna gain measurements which were separately performed. In all cases the measured received signal strength was well within the measurement uncertainty of the value predicted by theory. Numbers within 3 dB of theoretical were always achieved even on the longest paths.

One of the results of these measurements was an emphasis on the importance of antenna mounting. In order to achieve expected antenna gains, the 900 MHz yagi antennas need to be positioned at least 10-15 feet above ground and nearby clutter such as roofs and trees. If this is not done, the main lobe can be deflected upward with a resultant reduction in gain in the direction of the horizon. The vertically polarized colinear needs to be similarly located and must not have clutter in its vicinity if best performance is to be obtained. improper antenna mounting can easily result in the loss of several dB in received C/N.

Site Selection

Probably the most important lesson learned is the importance of locating antennas and sites to allow completely LOS paths. We found that even mild suburban clutter; trees, bushes and buildings, can degrade signal strength by 20 dB or more. Worse yet, it can contribute greatly to multipath distortion caused by differing wave arrival times which can create time jitter of the data edge and ultimately limit the maximum signalling rate (baud). With indirect paths we observed peak-peak variation of as much as 500 nanoseconds. These numbers are very much in agreement



with published reports pertaining to cellular telephone systems operating between 800 MHz and 900 MHz^[7].

Figure 1 illustrates the importance of obtaining a completely LOS path for each link. The effects translate directly to out-of-pocket expense for the user. It also indicates the importance of careful selection of the Hubmaster site. While the hub doesn't have to be and shouldn't

Probably the most important lesson learned is the importance of locating antennas and sites to allow completely LOS paths.

be high level, it should be LOS to each of the anticipated users. It is not necessary that the hub be at the

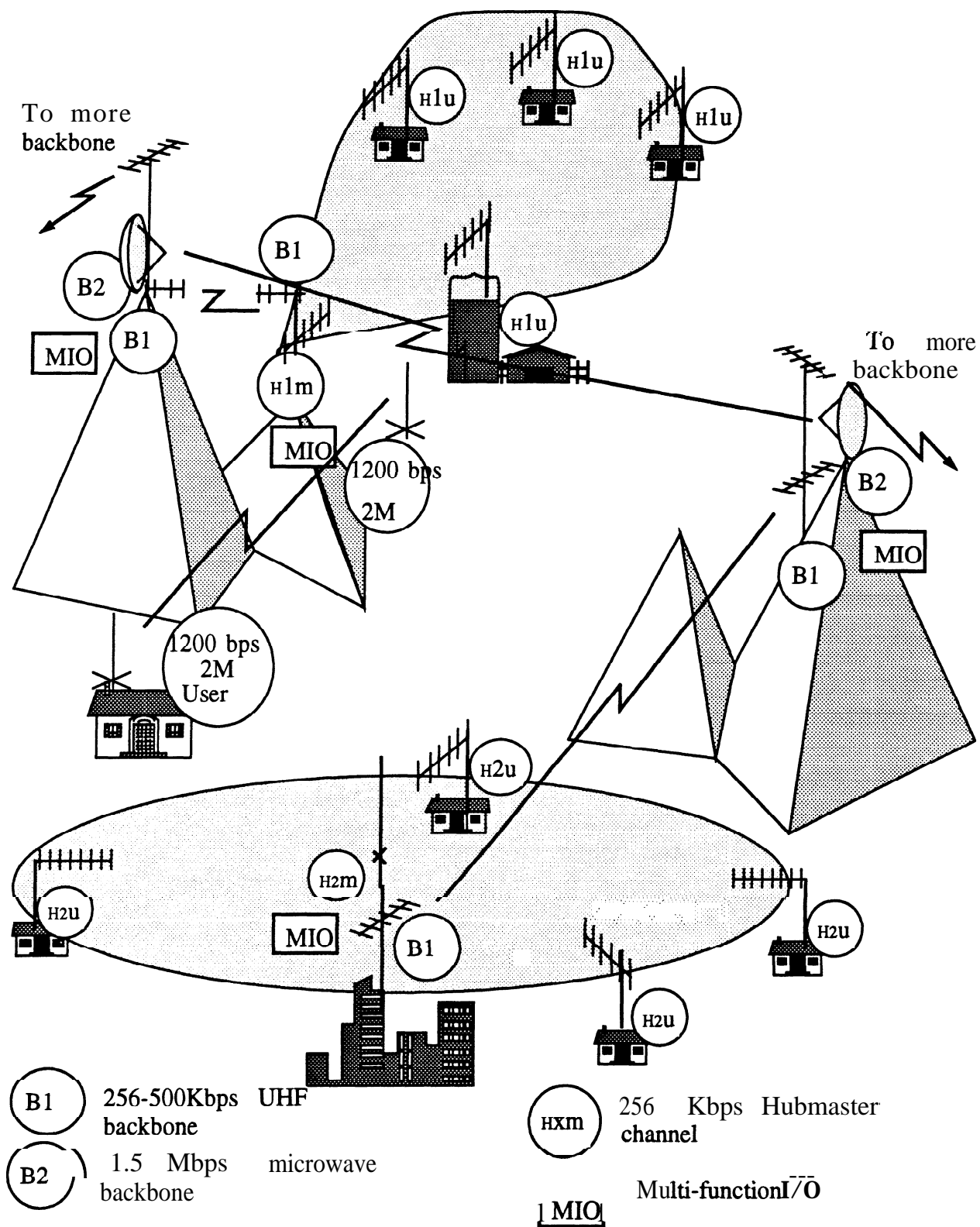


Figure 2. Common backbone and both high and low performance user access is provided in this example. LOS paths and directional antennas provide high performance with relatively inexpensive hardware.

physical center of a group of users. A very viable alternative is to locate it outside the cluster of users and use an antenna with directivity in both azimuth and elevation instead of the colinear with its omni-pancake pattern. This alternative is shown in cluster 1 of Figure 2. The hub directional antenna should be chosen with a beamwidth just sufficient to include the cluster members.

It is difficult to overstress the importance of carefully selecting the path and site although the results of not doing so are readily apparent. For a well located hub, tenth-watt radios appear to be very satisfactory for all users. This can result in excellent network performance with plenty of margin at minimum cost. If indirect paths are attempted even 10 watt or bigger transmitters may be insufficient and the accompanying distortion and path variations may make the resulting performance far from satisfactory even if higher gain antennas are used.

Getting good results from this higher speed network hardware will require special attention to the installation at each amateur's station. Instead of simply connecting a groundplane mounted on a window sill to an HT, as is commonly done now on the 2M band, the user will need to find a location with an unobstructed view towards the cluster server and mount a directional antenna. The directional antennas we have been using are homebrew yagis about 5 feet long but commercial antennas of similar performance should be just fine also. By working together with the other members of the local cluster to find a server site which is LOS to all members' antennas even inexpensive 100 milliwatt digital radios should be able to provide excellent

results. Generally, mounting the antenna at least 10-15 feet above the ground and roof and in a position where it can "see" the hub server antenna combined with using good quality feedline like Belden 9913 works fine.

By keeping the total number of users of the cluster small, perhaps 6 - 10, each user can have a guarantee of high data rate and low delays for his communication. As other local amateurs see exciting new applications which higher speed networking can offer they may be encouraged to work together to add additional clusters. Adding well situated clusters instead of adding users to existing clusters can allow everyone to experience the same excellent service at the lowest cost. Adding a cluster does require additional hardware for the server but this cost can be shared among the users in the same way that repeater clubs share hardware expense. The benefits of maintaining small cluster size by limiting the maximum number of users and choosing a server site which is completely LOS to each member far outweigh the slight additional per-user cost of the extra server hardware.

Elevation Profiles

Site selection is just as important for inter-cluster (backbone) communications. In seeking optimum sites for backbone hardware we spent many hours with topographic maps trying to determine good potential sites. After much consideration and reducing the possibilities down to a small number of likely candidates an extremely useful tool was made available by Rich Biby of Communications Data Services Inc. He has made an elevation profile database available to radio amateurs for non-

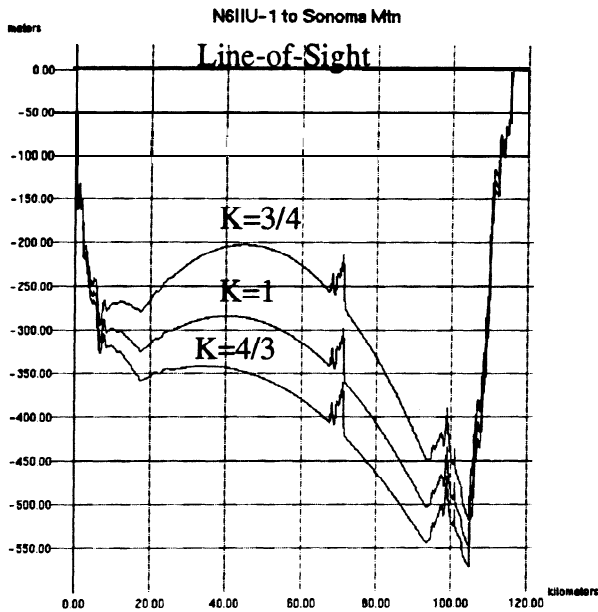


Figure 3. Elevation profile plot of a proposed N-S backbone path across the Golden Gate.

commercial use. We would like to thank him for the use of this database since we were able to very rapidly discover that several of the candidate paths and sites were in fact obstructed and not LOS. This prevented the expenditure of what would have been a lot of unproductive effort. Upon re-examination, the obstructions were visible on the topographic maps we used but were easily overlooked.

A graph of a path which did prove useful and was measured is shown in Figure 3. Three different K factors representing different effective earth radii are shown. Since this particular path is across the Golden Gate, the entrance to San Francisco Bay, it has moist marine air flowing in and out on a daily basis and may be a very difficult radio path. This movement can alter the refractive index and “de-steer” radio waves. For this reason earth curvature greater than actual ($K < 1$) is included. The $K=4/3$ value is a typical value to

account for the nominal radio horizon being greater than the optical horizon. This is one of the paths actually measured and appears to have enough vertical clearance to promise communication unobstructed by terrain even in the worst of situations. By situating the ends at similar elevations we hope to minimize problems due to “layering” of air masses.

Spectrum Assignment

Since higher performance networking requires greater channel bandwidth, spectrum sharing is another form of necessary coordination. In Northern California the Northern California Packet Association is involved in coordinating amateur spectrum for packet use. By indicating our expected needs to the NCPA ahead of time, a few wider channels were allocated for higher speed networking use in both the 900 and 1200 MHz bands.

Modulation and Data Encoding

Because high speed digital radios are not presently common there hasn't yet been a great need to coordinate modulation schemes with other users. Although the radios we are presently building use FSK and operate at 256 Kbps, selection of modulation methods and coding schemes which are most economical and immune to multipath distortion is an area needing much further work. Certainly as the network grows it will be desirable for different radio hardware to be compatible.

In the same way that setting current NBFM radio deviation and frequency is important to achieve good performance it is also necessary that modulation

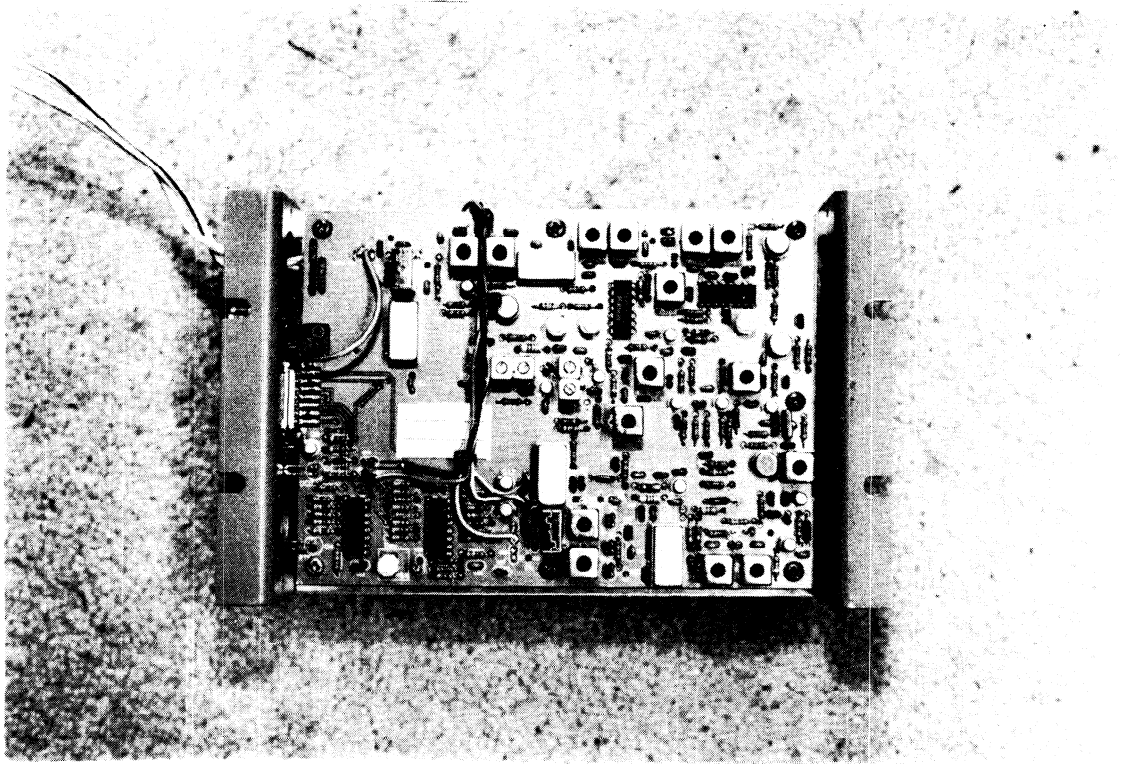


Photo 1. The main board of one of the 10 watt 256 Kbps 904 MHz transceivers

levels and adjacent channel transmitter levels be controlled so that everyone can make good use of the amateur bands used for the network.

The present radios are being designed to fit into a 1 MHz channel with signals outside the band approximately 30 dB or more below the carrier. By operating the cluster and backbone radios with opposite antenna polarizations as well as using directional antennas, much greater isolation between channels is possible than would be afforded with only omnidirectional antennas.

Since prototyping and building two 10 watt 904.5 MHz digital radios with a "garage printed circuit board process", twenty five sets of main and power amplifier boards have been commercially fabricated. Two 10 watt radios and two low power radios have

been built from these newest boards. Several more 10 watt radios are currently in the process of construction and testing. A photograph of one of the 10 watt radios is shown in Photo 1.

The most recent radios have been redesigned slightly for operation in a 1 MHz instead of a 2 MHz wide channel using a lower first I-F at 29 MHz. They show good performance, BER below 1×10^{-6} , for signal levels greater than about -90 dBm.

Link Layer

As previously indicated, Hubmaster is a protocol designed as a solution to the problem of "efficiently sharing a common channel by users using directional antennas and requiring LOS only to the hub. It is designed to be a useful solution for moderate speed

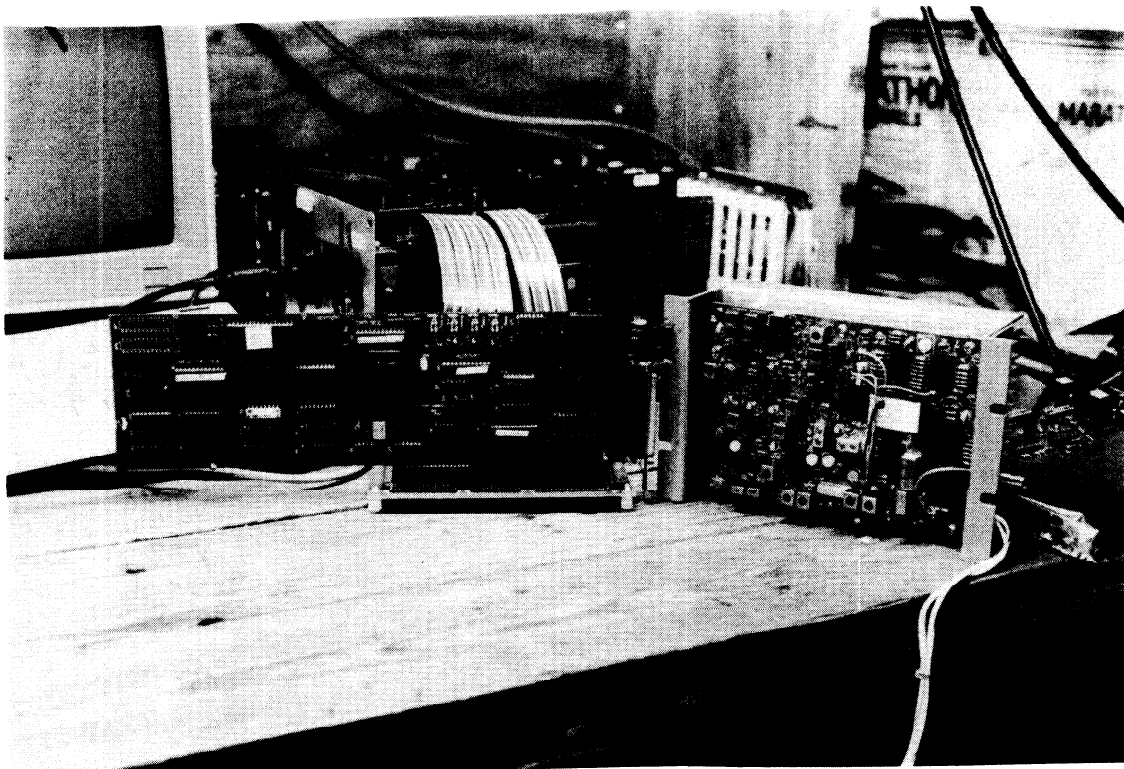


Photo 2. Multifunction I/O, MIO, on the left with a 10 watt radio. The PC in the background is being used for software development.

networking, not as the only answer. Hubmaster is intended to give users equal access to the network and to provide throughput and latency performance sufficient to support a wide variety of amateur networking applications.

As amateurs seek even higher data communications speeds we anticipate a different link architecture and protocol to support full point-point physical layer hardware.

In order to develop the Hubmaster protocol as well as the simpler point-point communication for the backbone links we needed radios and antennas but also the digital hardware to control them. MIO is the platform we developed to provide this.

MIO features.

Multi-Function I/O adapter, or MIO, was developed to provide the digital interfacing. MIO serves in the traditional TNC role of accepting data, getting them into the right link layer format and transmitted.

MIO is implemented as a full length ISA bus plug in card. MIO is actually a complete microcomputer system on a card including a NEC V40 microprocessor running at 8MHz, 768K of DRAM, two Enhanced Serial Communications Controllers (ESCC), up to 256K of EPROM (32 pin), and a shared memory interface to the ISA bus.

Programming and debugging software for MIO is done with the Borland BC++ V2.0 compiler and tools. These tools normally run on a PC and is the development environment presently

in place for much of today's amateur radio software packages.

MIO will also function in a stand-alone or "hilltop" mode. The only necessary dependency MIO has on the ISA bus slot is for +5 VDC power. By supplying power and loading software into the EPROM, MIO can serve as a hilltop isolated packet switch. Software can also be loaded into DRAM over the air.

The serial ports in MIO consist of a pair of Zilog 85230 ESCCs. The 85230 is the next generation of the 8530 used in present TNCs. Four serial data channels are provided. Three of them are connected via direct memory access (DMA) to the V40 microprocessor (μ P). One channel can provide full duplex at T1 (1.5 Mbps) serial data rates, two more can run half-duplex 256Kbps while the fourth is running in interrupt mode at 56Kbps.

The ESCC can provide full autovector capability to the V40 with hardware interrupt acknowledge. In most usages of SCCs, the device sends an interrupt to the μ P which then begins to check each source of interrupt within the SCC to determine the cause. Autovectored interrupts allow the interrupt service routine to be specific to the cause of the interrupt. This saves machine instructions in interrupt processing allowing higher through-put at less overall system cost.

DMA data transfer can be programmed for three of the ESCC channels. In systems lacking DMA, transfers would be done via μ P read/write to ESCC registers in an interrupt service routine. Hundreds of bus cycles would be required per byte. With DMA, only a few bus cycles are

needed per byte transferred.

The connection to MIO for serial data is via a DB37 female connector on the end of the card. TTL logic levels are used. Normally TTL would not be a good choice for interfacing long, highspeed lines. However, this system is still in the experimentation phase and TTL used with external interface logic affords the most flexibility. +12 VDC at 500 ma from the PC bus is also available on the DB37 for powering interface conversion circuitry.

To date, interfacing has been built to support differential ECL to the microwave^[5], and 900 MHz radios, an AM7910 for 1.2Kbps AFSK and high speed asynchronous serial RS232C for remote debugging.

The remainder of the logic on MIO consists of four PALs and eight MSI size gates all mounted on a four layer commercially produced PCB. It consumes 2.5 watts of power, similar to present TNCs.

The interface with the host PC is via one I/O-port control register, interrupts, and a shared memory window. The address of the shared memory window is set via the I/O-port control register. The size of the shared memory window can be set to either 8K or 64K. Future versions will be restricted to 8K. The idea of a 64K window was to minimize the move-in/move-out nature of traffic, but MIO moves data across the bus at 97% of memory transfer rates, so that an extra copy was not too much of a performance problem relative to the line speeds of the serial ports. The I/O-port address is set via on-board jumpers.

The NEC V40 is an Intel-style architecture μ P. Built into the μ P are four DMA channels, an interrupt controller,

three timers, bus interfacing logic, DRAM refresh, and external bus master control.

The **V40** allows programming from a familiar environment with an existing set of tools. A PC can be used for development and code can be written in Borland C++. Stuart Phillips, **N6TTO**, in his usual get-the-job-done efficiency has developed a collection of tools which make programming and debugging **MIO** a dream^[6]. First, one writes normal embedded C code with **BC++**. Then the code is **TLINKed** with a custom version of the **BC++** startup file "CO". **TLINK** produces the usual .exe file. The file is in normal .EXE format, with relocation items.

At this point **N6TTO**'s utilities really start to come into play. **COMF** takes the .EXE file produced by **TLINK**, fixes up all the relocation items and produces a .LOD file. The code file is now a memory image and is ready for loading onto **MIO**. Next a loader is supplied which will access the shared memory interface and load the file onto **MIO** so that execution can begin.

Debugging in an embedded environment is always a bit of a chore, since watching what happens can take expensive instruments. Again, **N6TTO** took on the task of developing special tools for **MIO**. **BC++** is supplied with a graphic, screen oriented debugger called Turbo Debugger (TD). TD allows the developer to single step through a program in either assembler or C source lines. Break points can be planted for trapping. At the same time, the **TD** window displays the CPU registers, and data portions. Variables can be examined by name.

Borland supplies a version of **TD**

called **TDREMOTE** that allows the target program to run in one **PC** while the debugger runs in another **PC**. The two PC's are linked via serial COM ports. **N6TTO**'s **MIOTDREM** affords the same capabilities for debugging software on **MIO**.

A typical scenario would be **MIO** serving as a controlling point of a Hubmaster site. One serial port is connected to a **900MHz** radio used to 'poll' secondaries (users) in the cluster. Another port is hooked to a full-duplex microwave link (backbone) to other clusters. The third port could either serve as a connection to other clusters, or as a gateway port to **2M 1.2Kbps** AFSK or a **56Kbps WA4DSY** setup. The fourth port might be tied to a phone or leased line for non-RF access.

In this situation, the software on **MIO** would use the Hubmaster protocol to move traffic among the secondaries and to and from other clusters ^[4] as illustrated in Figure 2.

Progress to Date

To date, we've been able to get packets moving across the bench top by way of RF. On command from the PC, packets are sent out **MIO** via the **ESCCs** to a **TTL/ECL** interface and into one of the **904 MHz** radios. Incoming packets from a second radio traverse the reverse of this setup and are examined on the target **PC**, which is the same **PC** as the source. This is simply back-to-back packets and something one could do with a null modem cable. In fact, that's how initial **MIO** testing started. Although this is a short RF path across a garage, signal strength and **BER** measurements over many different real paths give us confidence that these

results will apply well to final installations into a network. During the first testing, software coding errors had the 256 Kbps radios running at something higher than the intended data rate; data was flowing at 624 Kbps!

At the time we write this the next task at hand is to reproduce this same test across a South Bay/North Bay RF path, with the data rate reduced to 256Kbps.

N6PLO is building a different I/O controller that will connect to any host with an ethernet port. Using a Zilog 16C50 DDPLL for decoding and clock recovery, he has developed an adapter for non-SCC serial interfaces, such as those provided by the Motorola 68302. Another short term goal is to demonstrate the interoperability of the two systems.

Network Layer

Amateur radio doesn't presently have an operational network layer. If amateur radio digital communications is to progress, a network layer needs to emerge. In this paper, we've detailed our progress in solving some of the physical and link layer issues. We've stressed the need for coordination and the direction technology must go to gain improvements in data throughput.

A network layer provides the "reachability" between two end points and the facility to extend communication beyond a single RF path. A network layer, if it's to serve the needs of all, must be transparent to the protocols above it, and needs to be able to maintain end-to-end "reachability", independent of the actions of the two communicating end-points.

The direction of amateur networking today is very much application specific.

Operation is over connected mode AX.25. AX.25 is a link layer protocol which has been extended to function as a network and transport protocol all at the same time.

AX.25 is also a source routing protocol. Source routing presents the problem that the end points must get directly involved in how packets flow from end to end. Each end point must be prepared to deal with the dynamics of packet flow, including loss of various channels, congestion in a path, and path delay variances. Route discovery is complex as well. Multiple paths may exist between two end points. Which path is the best? Source routing makes end point nodes overly complex to manage and implement. This costs the end user money and performance.

AX.25 also lacks the necessary throughput algorithms to provide consistent service on faster links and diverse network paths. The AX.25 HDLC style go-back-N scheme doesn't have enough sequence space to keep faster pipes full. AX.25 could easily cause a 256Kbps channel to go as slow as today's 56Kbps or 1200 bps channels.

This paper is a progress report. Our previous papers give direction to a network layer. If you've read this far, you know we're barely past making link layer communications happen. The real point of our experiments is to provide amateur radio with a means to build a network layer. MIO is the first step toward a network layer, layer three (L3), TNC.

A L3 TNC resides at the users location and serves as the connection point for end to end "reachability". The L3 TNC accepts packets containing transparent data and a destination

address. It then uses the Hubmaster protocol to get the packet sent to the master. The master's routing table consists of entries with destinations or supersets of destinations, the next hop to the destination from this master and the number of hops from this master to the

Our desire is for users to view the network as a means of improving the quality and increasing the number of types of communication possible among all amateurs.

destination. When the layer 3 TNC frame arrives at the master this table is used to properly forward the packet.

Only master stations get involved in determining routes. The end point L3 TNC *and the users application* do not need to process routing information. Network operation will be transparent to the manner in which user applications access the network. User applications can be written without having to be tailored for each local network style.

Coordination Among Users

Perhaps the most difficult aspect of building a true amateur network relates to finding ways for the great number and diversity of amateurs to work together to achieve a common goal. In order for this common effort to occur, the many participants must be able to perceive and experience the benefits of network access. Each contributor must find a gain in his or her own area of interest.

Our desire is for users to view the

network as a means of improving the quality and increasing the number of types of communication possible among all amateurs. The user's means of access, whether with conventional 1200 bps AFSK hardware or with much higher data rates supported with Hubmaster or other protocols, should not be connected with a particular service or application. Different amateurs

interested in emergency communication, DX spotting sub-networks, experimenting with digital audio and those interested only in conventional electronic mail and BBS functions should each be able to pursue his own interests through identical connections to the network.

It is our desire that the connection of application with amateur band or frequency be eliminated. No longer would there need to be different frequencies for BBS traffic and access, keyboard-keyboard QSO and network experimentation. This will allow improvements in the network to be experienced by all. If a faster backbone is put into place, everyone can experience better performance. If previously unconnected subnetworks are bridged then everyone finds that their region of communication is increased. This commonality of resource provides the incentive for common effort towards implementation, support and improvement. Certainly for some applications higher performance user access will be required and higher frequencies will be necessary to support this but channelization based only on application can be eliminated. The idea of a channel 'closed' to all but one kind of application can disappear.

Neither should the network be viewed as a particular set of protocols.

Our desire is for an amateur network first not a TCP/IP or other particular protocol network. Nor is such a network just for “computerized” amateurs. We hope for information communications of all kinds, traditional and new, which can offer new relevance to all amateurs and new life and growth to the hobby as a whole.

References

- [1] *The Implications of High Speed RF Networking*, Mike Chepponis, K3MC, Glenn Elmore, N6GN, Bdale Garbee, N3EUA, Phil Karn, KA9q, and Kevin Rowett, NGRCE, 8th ARRL CNC, 1989
- [2] *Physical Layer Considerations in Building a High Speed Amateur Radio Network* Glenn Elmore, N6GN, 9th ARRL CNC, 1990
- [3] *Implementation of a 1 Mbps Packet Data Link*, Glenn Elmore, N6GN, and Kevin Rowett, NGRCE, 8th ARRL CNC, 1989
- [4] *Hubmaster: Cluster-Based Access to High-Speed Networks*, Glenn Elmore, N6GN, Kevin Rowett, NGRCE, and Ed Satterthwaite, N6PLO, 9th ARRL CNC, 1990
- [5] *A 2-Mbit/s Microwave Data Link*, ARRL Handbook for Radio Amateurs, 1991, 32-65
- [6] *C++ for Embedded Systems*, Stuart Phillips, N6TTO, Dr.Dobb's Journal, 1991
- [7] *IEEE Journal on Selected Area in Communications*, Vol. SAC-5, No. 2, February 1987

Distributed Directory Services for the Amateur Packet Radio Network

Andrew Funk, KB7UV
14-23 31st Avenue . Apt. 4A
Astoria, NY 11106-4559 USA
kb7uv@kd6th.nj.usa (Packet)
74756.2055@compuserve.com (Internet)
The Ratio Amateur Telecommunications Society

The Amateur Radio Packet Network has implemented an ad-hoc, hierarchical, area-based message addressing system. In the United States these areas resolve down to states or local domain. Packet Bulletin Board Systems (PBBSs) also support a limited user directory service, implemented as the “home BBS” function. This sets the stage for relatively easily implementing a distributed directory service for the packet network.

The Problem

It isn't easy to ensure that a packet message will reach its intended recipient. There is no universal, automated system to keep track of which Amateurs are receiving their packet “mail” at which PBBS. (Yes, there are White Pages servers, but their operation is neither universal nor automatic.) Originators must both know and explicitly specify the destination PBBS when sending a message if it is to have a reasonable chance of being received.

There are exceptions which at times make it easier to send packet messages. PBBSs can automatically address messages when they “know” the “home BBS” of the

addressee. Some systems can query White Pages servers to determine the correct destination PBBS. However, these are exceptions to the situation. facing most packet users.

A Proposed Solution

Consider the situation if all PBBSs “know” where to send messages addressed to all active packet operators in their area. Amateurs could then easily correspond with others in their area without having to know which system they frequent. If one wanted to send a message to an Amateur in a distant state or province the “address” need only consist of the destination state or province; upon reaching the first PBBS in the destination state or province it would be re-addressed to the correct PBBS.

This can be accomplished with a modification to the “home BBS” function of Packet Bulletin Board System software and establishment of state/province (or sub-area) area “flood” message distribution.

As currently implemented the “home BBS” function requires manual updates, and operates in what seems to be a “backwards” manner. Users must connect with as many PBBSs as possible and inform these systems of their preferred “home BBS.” If users change their preferred “home” they must again connect to each system to update the information.

It would be much more logical for users to connect to their system of choice and declare it to be their “home BBS.” PBBSs would share “home b’bs” data within their area. All systems within the state/province, or sub-area, would know the “home bbs” of all active packet operators in their area.

Consideration must be given to dealing with packet operators who move from one area to another. As part of the procedure to declare a PBBS as one’s “home,” the user must be asked for the callsign of their previous “home BBS.” The previous “home,” and all systems in its area, must also be advised of the new “home BBS” declaration.

Another special case is temporary operation. Many Amateurs travel. Some live in different areas throughout the year. This requires the ability to declare a system to be one's temporary "home BBS."

Update messages should be allowed to contain multiple updates, but each individual entry must be date/time-stamped. This will permit checking that update information is truly new, and not an old message which was delayed or re-sent for some reason. In addition there should be fields for optional inclusion of effective and expiration dates. This will provide for advanced scheduling of "home BBS" changes for "migratory" users.

Proposed PBBS Dialogues

The following is a proposed dialogue for PBBSs to use for the new implementation of "home BBS":

```
@KB7UV MailBox>
```

```
home
```

```
Your home BBS is the PBBS where you will receive yourpacketmail.  
If you choose this as your home BBS system*ALL* packetmessages  
for you will be routedhere. You can only have one home BBS.
```

```
Your current home BBS is WB2GTX.
```

```
{Youdonotcurrentlyhave a home BBS.}'1
```

```
Change home BBS to KB7UV? [Y/N/?]
```

```
Y
```

```
This is now your home BBS. All yourpacketmailwillbe forwarded  
to this system.
```

```
@KB7UV MailBox>
```

¹If there was no "home bbs" information on file for the user.

For both temporary operation and permanent changes to other areas, Amateurs who know which system will become their new home (permanent or temporary) should be able to inform their current home system of the change. This dialogue could be similar to this:

```
@KB7UV MailBox>
```

```
nothome
```

```
{This system is not your "home" BBS -- cannotprocess "nothome"
request. @KB7UV MailBox>}2
```

```
You are currently receiving yourmailhere. Do youwishto change
to another BBS [Y/N] ?
```

```
Y
```

```
What is the callsign of the BBS you wish to change to?
```

```
W1AW
```

```
{W1AW does not appear in the "known BBS" list. In what
state/province (Z-letters) is it located?}3
```

```
You have chosen W1AW.CT.USA as your new home BBS. Is this correct
[Y/N]?
```

```
Y
```

```
Your home BBS change to W1AW.CT.USA has been accepted. W1AW and
all BBSS in CT will be informed of yourmove.
```

```
@KB7UV MailBox>
```

Message Routing

²This would be the response if the user attempted to use this command on a PBBS other than their "home bbs."

³This prompt would be sent if the specified new "home" was not "known" to the PBBS. The state is needed in order to route the automatically generated update message to the specified new "home" system. Following the query for state the PBBS should ask for the country of the new "home BBS."

Whenever a message arrives at a PBBS the system will first look to see if it already fully addressed for another destination PBBS. If so the addressing will be left alone, and the message will be forwarded out based upon that address. This ensures that messages fully addressed upon origination will not be “tampered” with.

If an arriving, or originated, message is not fully addressed whatever routing information it does carry will be checked. If it is for another area it will be forwarded as specified. If it is for the same area as the PBBS, or if it has no routing information, the PBBS will check to see if the addressee is “known.” For “known” addressees the system will either readdress and forward the message, or hold it for pick-up, as appropriate. Messages for “unknown” addressees without routing information are more of a challenge.

The first check of these “unknown” messages should be of the callsign prefix of the addressee. Foreign callsigns are rather easy: route the message to the appropriate country.⁴ Domestic callsigns can be automatically looked-up through a nearby White Pages server for routing information, if available, or through a callbook server to obtain the addressee’s state or province. Messages for which routing was found at the White Pages server will be sent as specified. Out-of-state/province messages, looked-up through a callbook server, should then be forwarded to the correct states or province. Messages with instate/province addressees which are not known to the White Pages Server, and those whose addresses could not be found at all, require special handling.

If there are sub-areas within the state/province, an inquiry should be sent to those other sub-areas to see if the addressee is known. If a reply is received with routing information the message should be re-addressed and forwarded. If no routing information is received within a specified time period (seven days seems reasonable) the message should be returned to the sender as “undeliverable, addressee unknown.”

⁴Many amateurs operate under reciprocal agreements. The six-character “To:” field does not provide for portable designations... This common PBBS software limitation must be rectified.

Why Not Dedicated Servers?

Others have suggested utilizing dedicated Directory Servers for the Amateur Packet Radio Network. These schemes would require address queries and responses for **each** message to be forwarded. Without higher speeds this will further burden our already heavily trafficked networks. Distributing directory information among message servers will limit these queries and responses to those messages destined for other areas.

This is not to say that dedicated Directory Servers wouldn't be helpful along with this proposal for distributed directory services. Dedicated servers in all areas, sharing information, would provide a more efficient means for dealing with unknown addressees.

Implementation

As current BBS software already supports the "home BBS" function, albeit in what seems to be a backwards implementation, it should be relatively easy for this distributed directory service proposal, or something similar, to be implemented. This paper outlines needs and a proposed solution; the actual implementation will need to be a cooperative effort by the PBBS software authors.

There is an additional requirement for this to work. Each state/province, and any sub-areas, must establish area flooding forward routes. All PBBS systems will need to be able to forward messages addressed simply as "@ΣT" where "CT" is a Z-letter state/province designation. In this way update messages can be sent to "UPDATE@ΣT," "WPAGES@ΣT," or similar for automatic distribution to all PBBSs and White Pages servers in the region.

Remote Operation Message Formats

. Home Declaration

header: SP UPDATE@ΣT <BBSCALL \$123456-BBSCALL

title: not used — blank or informative, such as: “New Users This System”

text: one or more line of the following:

“callsign YYMMDDHHMM [START-DATE EXP DATE]”

Where “YYMMDDHHMM” is a date/time stamp indicating when the user made the “home bbs” declaration, and **START_DATE** and **EXP DATE** are optional fields of the format **YYMMDD** for effective and expiration dates.

Note: The “home bbs” for the callsigns in the message is the originating PBBS as shown in the header. Use of RFC-822 headers is suggested so as to place this information explicitly in the body of the message.

. Callsign Look-up Request To Address Server

header: SP LOOKUP@*ServerCall* <BBSCALL
\$123456_BBSCALL

title: REQSTATE

text: one or more line of the following:
“call sign”

. Response From Address Server

header: SP DAEMON@BBSCALL <*ServerCall* \$34567_*ServerCall*

title: RE:REQ_STATE

text: one or more line of the following, as appropriate:
“Callsign Address”
“Callsign ***NOT FOUND***”

Where Address is in one of the forms:

“ΣT.cou”

“bbSCALL.ΣT.cou”

(“ΣT” — 2-letter state/province code)

SPECIFICATION OF THE AVC-R-ISA MAC LAYER PROTOCOL

A. Giordano (I1TD), A. Imovilli (IW1PVW), C. Nobile (IW1QAP),
G. Pederiva (IW1QAN), S. Zappatore (IW1PTR)
[al@dist.unige.it]

Dep. of Communication, Computer and Systems Science (IK1HLF)
University of Genoa
Via Opera Pia, 1 I/A - 16145 Genoa, Italy

Abstract

The paper deals with the AVC-R-ISA access protocol, developed in order to maximize the one-step throughput and to solve the hidden stations problems, in a packet radio network cellular structure.

After a brief description of the overall AVC-R-ISA strategy, the actual protocol machine is presented for both kinds of stations (master and slaves) that are presented in the network. A short description of the frame structure is also shown along with a brief discussion about preliminary real world performance results.

1. Introduction

The AVC-R-ISA protocol, proposed and described in [1], represents a strategy for the access rights distribution on a common channel in order to maximize the one-step aggregate throughput. The devised strategy is characterized by its adaptability to channel load variations that may be due to change in radio station density and/or packet generation rate. The presented multi-access protocol refers to a cellular network structure and needs a specialized station, in each cell, referred to in the following as master station or base station, that acts as a synchronization entity and as a feedback and status information disseminator, because of the previously mentioned tasks, the base station is constrained to be placed in a site from which it can directly communicate with any other station in the cell.

In the following, after a brief overview of the overall access strategy, the Medium Access Control Protocol (MACP) and the frame structure for AVC-R-ISA are described. In the last section the complete protocol machine is presented as it was implemented in the 910618 version of the Net/NOS package by Phil Karn.

2. The AVC-R-ISA strategy

The Access Virtual Channel-Radionet-Independent Stations Algorithm is a MAC layer protocol devoted to the solution of the multiple access problem for a packet-switching radio network. Its peculiarity is the capability to adapt to the traffic load variation due to changes both in the number of active stations and in the frame generation rate.

Futhermore, unlike from other MAC layer protocols (e.g. CSMA) the AVC-R-ISA performances are insensitive to the presence of “hidden” stations (i.e., stations that cannot establish one-hop radio links). As already mentioned, the only constraint of this protocol is the need to place the base station in a suitable site, from which it can communicate with any other (slave) station.

The AVC-R-ISA employs an informative centralized data structure (managed by the base station) along with some minor decentralized information.

The slave stations cooperate with the master according to the ISA [2] strategy, for the channel resource assignment. The ISA control rule is applied to a population of slave stations working on a slotted channel.

For each slot we know the channel status information, under the hypothesis that the frame presence in a station buffer is statistically independent of the state of the other slaves and that the station has an unitary frame buffer, so that any frame generated while the buffer is full is discarded.

The goal is to assign the right access to the stations, in order to maximize the one-step throughput and consequently, the probability of no-collision transmission. This is accomplished by uptating a vector $p(t)$ (whose components represent the marginal probability of the presence of the packet in a slave station) on the basis of the stations’ frames generation rate and of the channel feedback reporting one of the following three states : empty slot, collision or successful transmission.

A major problem for a radio slotted network is devising a simple and efficient synchronization mechanism: the AVC-R-ISA achieves this coordinated mechanism by having the base station and the slaves population synchronizing on their respective end-of-carrier (EOC) events [3]. The time interval beetwen two consecutive start-of-carriers by the master will be called “decisional interval” or “generalized slot”. The base station, besides acting as a synchronization entity, provides the broadcasting of feedback along with some global network information, as the slaves are not able, in general, to get this information directly.

The AVC-R-ISA control rule assigns access rights individually, and this requires that the stations possess a local identifier to be assigned upon entering a cell.

The dynamic assignment of local identifiers can be viewed as a “virtual channel number” assignment. The stations that want to enter the

network may use a special identifier, say 0, to ask the controller to release an actual identifier.

Thus the identifier number 0 is temporarily common to all users that have not yet been assigned their own channel number.

The incoming stations can generate a connection request by means of an AVC-R-ISA packet devoted to this goal, whenever channel 0 is enabled by the overall ISA algorithm. Obviously, conflict may occur among the stations while using this common identifier, even when channel 0 alone is enabled.

To resolve these conflicts, a controlled Aloha strategy, based on Rivest pseudo-Bayesian broadcast [4], is used. According to the latter, the master estimates the number of incoming stations and tunes channel 0 scheduling rate.

3. The AVC-R-ISA Medium Access Control Protocol (MACP)

The protocol related to the AVC-R-ISA strategy provides a method of establishing, maintaining and terminating the AVC-R-ISA connection. MACP goes through three distinct phases:

1) *Link establishment*

Before, any information may be exchanged, the protocol must first open the connection through an exchange of Connection packets. This exchange is complete, and the Connection state entered, once an Acknowledge Connection packet has been sent from the Master station to the Slave. Any non-AVC-R-ISA link establishment packets received before this change is completed, are discarded.

2) *Information exchange*

Until a link termination action occurs we have a regular information exchange among radio stations according to the AVC-R-ISA strategy.

3) *Link termination*

The protocol may terminate the link at any time. This usually can be done at the request of a human user or when the master detects a number of empty slots greater than a fixed threshold, for a given station.

MACP is specified by a number of packet formats and by a protocol machine. We now present an overview of the MACP automation, followed by a representation of the state transition table. As the master and the slave stations perform different tasks they need two different MACP automation.

There are three classes of MACP packets according to the previous mentioned phases:

1) Link establishment packets used to establish a link (e.g., Master/Slave connection request, Master acknowledge connection).

2) Information exchange packets, that encapsulate the upper layer information. The master station not only uses this kind of packets to transmit the upper layer data, but also to broadcast the MAC control

information (e.g., Master synchronization, Slave information). An exception is represented by the “Generation rate updating” packet, sent from the Slave, that comprises only AVC-R-ISA supervisory information.

3) Link termination packets used to terminate a link (e.g., Master/Slave disconnection request, Master acknowledge disconnection, Master rejected connection, Master shutdown).

The protocol machine is defined by events, state transitions and actions. Events include receipt of external commands such as Switch On/Off and *Disconnection*, expiration of the Connection/Disconnection/Synchronization timers, and receipt of packets. Actions include the starting and restarting of the timers, the transmissions of packets and updates of slave station empty slot counters in order to provide an automatic link termination.

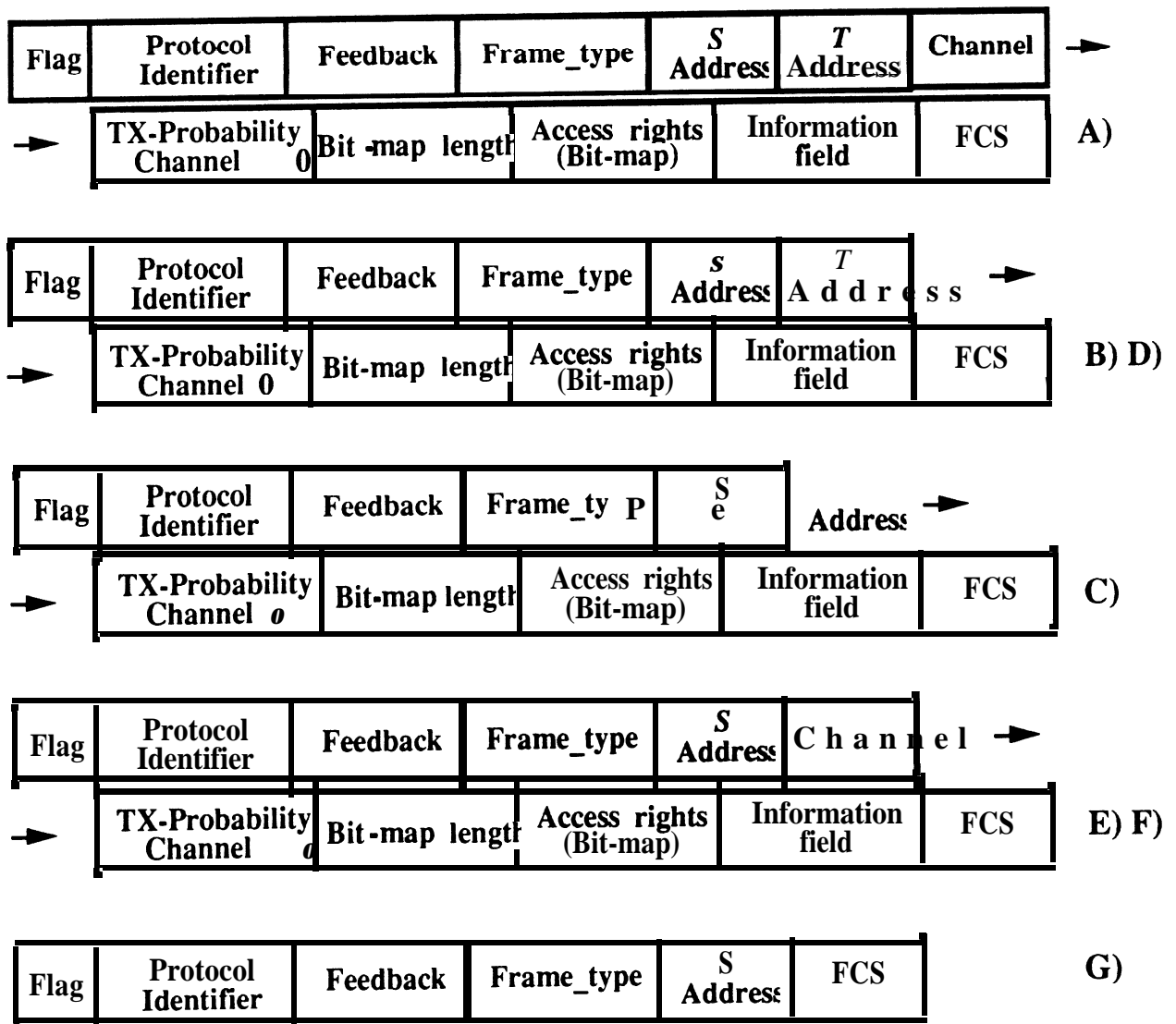


Fig. 1 S (Master Station)-FRAMES : A) *T* Connection Acknowledge; B) *T* Connection Request; C) Synchronization; D) *T* Connection Rejected; E) *T* Disconnection Request; F) *T* Disconnection Acknowledge; G) Shutdown (forced disconnection of all the *T* station).

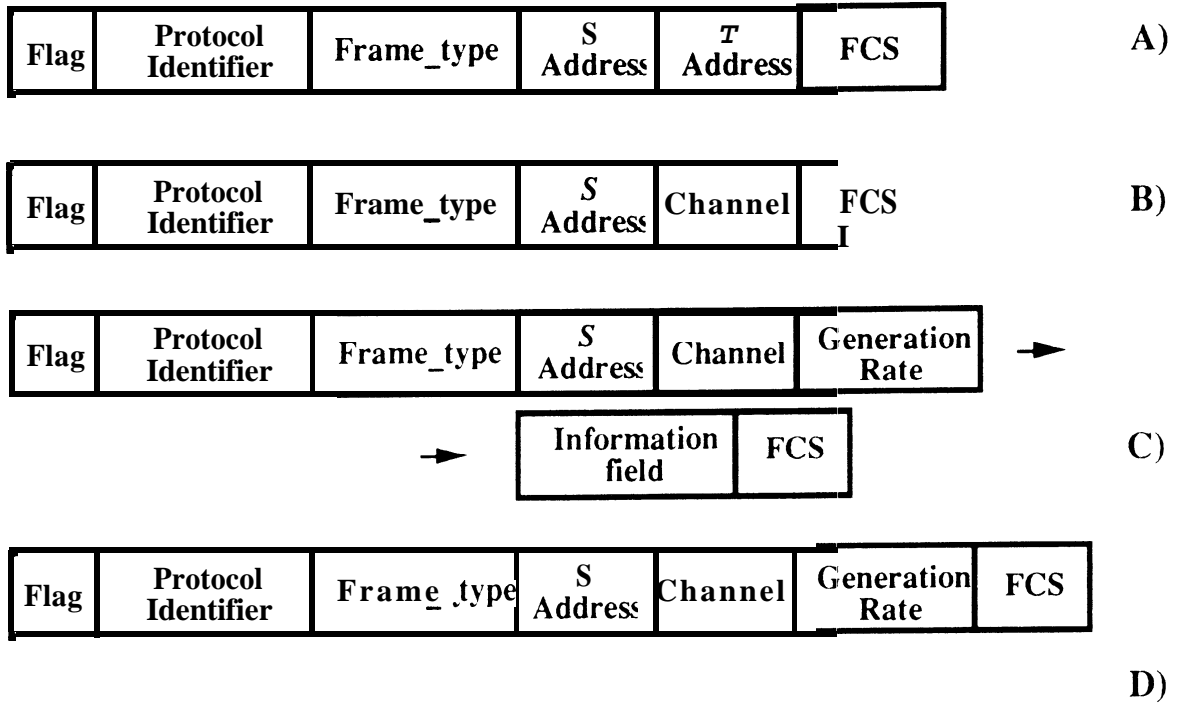


Fig. 2 S (Slave Station)-FRAMES : A) Connection Request; B) Disconnection Request; C) Information; D) Generation Rate Updating.

The complete set of MACP frames is depicted in Figs. 1 and 2 where the Master and Slave stations are indicated as S and T, respectively.

4. State Transition Tables

The complete state transition tables for the Master and the Slaves follow. States are indicated in the rows, and events are read in the columns. State transitions and actions are represented in the form action/new-state. Actions caused by the same event are represented as action1 &action2&....&actionN. Alternative choices are represented as action1|action2 or event1 |event2.

Master State Transition Table

Events	State		
	1	2	3
	I NON-ACTIVE	DISCONNECTED	CONNECTED
<hr/>			
ISA+		st+/2	2
ISA-	I	1	(st-&sshd)/1
STO	I	np/1	2
NFT		1	ssnc/2
RSCR	I	1	sokc/3
RSDR		1	2
RINF		1	2
RGRU		1	2
RMCR		1	2
RMDR		1	2
ROKC	I	1	2
ROKD		1	2
RRJC		1	2
RSHD		1	2
RSNC		1	2

(*) *ssnc* in the normal transmission, **smdr** in case a Slave reached the maximum allowed of empty slot. The transition to state 2 takes place only incase an **smdr** is transmitted and there is only one station connected.

(**) sokc in a normal transition, srjc in the case the maximun number of slave station allowed within the cell is reached.

(***) Goes to state 2 only if *sokd* is sent and there is only one station connected.

In the following we briefly describe the meaning of the previous mentioned states, events and actions.

States

- NON-ACTIVE The Master station cannot do activity on the network;
- DISCONNECTED The Master station is active but no one Slave is connected;
- CONNECTED At least one Slave is connected to the Master.

Events

- ISA+ Related to “ISA switch on” user command [5];
- ISA- Related to “ISA switch off” user command;
- STO Synchronization timeout;

- NFT New Frame Transmitted (a new frame is inserted in the transmission queue);
- RSCR Received Slave Connection Request;
- RSDR Received Slave Disconnection Request;
- RINF Received Information for Slave;
- RGRU Received Generation Rate Updating;
- RMCR Received Master Connection Request;;
- RMDR Received Master Disconnection Request;;
- ROKC Received Acknowledge Connection;
- ROKD Received Acknowledge Disconnection;
- RRJC Received Rejected Connection;
- RSHD Received Shutdown;
- RSNC Received Synchronization.

Actions

- smcr Sent Master Connection Request;
- smdr Sent Master Disconnection Request;
- sokc Sent Acknowledge Connection;
- sokd Sent Acknowledge Disconnection;
- srjc Sent Rejected Connection;
- sshd Sent Shutdown;
- ssnc Sent Synchronization;
- st+ Synchronization timer start;
- st++ Synchronization timer restart;
- st- Synchronization timer stop;
- sc+ Counter slot increment;
- sc- Counter slot reset for transmitter Slave;
- np Not possible (timer is stopped!).

Slave State Transition Table

Events	I State					
	1	2	3	4	5	6
	NON-ACT.	DISC	WAIT-CN	CONN	WAIT-DS	TRANS.
ISA+	2	2	3	4	5	6
ISA-	1	1	ct-/1	dt-/6	6	1
DIC	1	2	ct-/2	dt+/4	5	6
CTO	np/1	np/2	A*	np/4	np/5	np/6
DTO	np/1	np/2	np/3	dt-/5	np/5	np/6
NTO	np/1	np/2	nt-/2	nt-/2	nt-/2	nt-/1
NFT	1	(ct++&nt+)/3	3	4**	5	6
GRX	np/1	np/2	np/3	4***	5	6
RSCR	1	2	3	4	5	6
RSDR	1	2	3	4	5	6
RINF	1	2	3	4	5	6
RGRU	1	2	3	4	5	6
RMCR	I 1	ct+/3	sscr/3	B	ssdr/5	ssdr/6
RMDR	1	2	ct-/2	dt-/2	2	1
ROKC	1	2	ct-/4	B	ssdr/5	ssdr/6
ROKD	I 1	2	3	B	2	1
RRJC	1	2	ct-/2	B	ssdr/5	ssdr/6
RSHD	1	2	ct-/2	dt-/2	2	1
RSNC	1	2	ct-&sscr/3	B****	ssdr/5	ssdr/6

where A ((ct++&sscr)/3)|(ct-/2)
 B (sgrulsinflna)/4

(#) The timer signaling the no activity by the Master station is restarted upon reception of a valid packet in states 3, 4, 5 e 6.

(*) The first timeout of the connection timer yields (ct++ & sscr)/3 whereas the second ct-/2.

(**) The frame queuing for transmission and it is sent when the enabling signal from the master is receiving.

(***) The action is activated by sgru, when the enabling signal from the master is receiving.

(****) The choice is made according to grx/nft/(empty queue) respectively.

States, events and actions related to Slave stations are described in the following.

States

- NON-ACTIVE The Slave station cannot do activity on the network;
- DISCONNECTED The Slave station is active but it is not connected;
- WAIT-CONN The Slave station is waiting for connection and to receive its local identifier;
- CONNECTED The Slave is connected to the Master;
- WAIT-DISC The Slave is waiting for disconnection; the final state upon receiving Acknowledge Disconnection is DISCONNECTED;
- TRANSITION The Slave is waiting for disconnection; the final state upon receiving Acknowledge Disconnection is NON-ACTIVE.

Events

- ISA+ Related to “ISA switch on” user command;
- ISA- Related to “ISA switch off” user command;
- DIC Related to “ISA disconnect” user command;
- CTO Connection timeout;
- DTO Disconnection timeout;
- NFT New Frame Transmitted (a new frame is inserted in the transmission queue);
- GRX Generation Rate Variation exceeding threshold;
- RSCR Received Slave Connection Request;
- RSDR Received Slave Disconnection Request;
- RINF Received Information for Slave;
- RGRU Received Generation Rate Updating;
- RMCR Received Master Connection Request;
- RMDR Received Master Disconnection Request;
- ROKC Received Acknowledge Connection;
- ROKD Received Acknowledge Disconnection;
- RRJC Received Rejected Connection;
- RSHD Received Shutdown;
- RSNC Received Synchronization.

Actions

- sscr Sent Slave Connection Request;
- ssdr Sent Slave Disconnection Request;
- sinf Sent Information;
- sgru Sent Generation Rate Updating;
- ct+ Connection timer start;
- ct++ Connection timer restart;
- ct- Connection timer stop;
- dt+ Disconnection timer start;
- dt- Disconnection timer stop;
- na No action (transmission queue is empty!);
- np Not possible (timer is stopped!).

5. Conclusions

We have presented in some detail the state transition tables of the AVC-R-ISA MAC protocol. AVC-R-ISA is actually running as MAC access protocol on a network covering north west part of Italy. The network is the test-bed for the connection of local networks. Slave stations act as local area routers while the master station, well positioned on a 5,000 ft top, manages the half-duplex channel, on the 70 cm band, according to the ISA algorithm. AVC-R-ISA is on the air since half year and this protocol machine represents the last version of an accurate and necessary tuning effort.

References

- [1] F. Davoli, A. Giordano, S. Zappatore, "Architecture and protocol design for a car-to-infrastructure packet radio network", *Proc. IEEE, GLOBECOM '90*, S. Diego, CA, Dec. 1990, pp. 1579-1585.
- [2] M. Aicardi, G. Casalino, F. Davoli, "Independent stations algorithm for the maximizations of one-step throughput in a multiaccess channel", *IEEE Trans. Comm.*, vol. COM-35, pp. 795-800, Aug. 1987.
- [3] M. Aicardi, F. Davoli, A. Giordano, S. Zappatore, "An adaptative multiple access protocol for a packet radio network of mobile and fixed stations", *Proc. 2nd PROMETHEUS Workshop*, Stockholm, Sweden, Oct. 1989, pp. 378-386.
- [4] R.L. Rivest, "Network control by Bayesian broadcast", *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 323-328, May 1987.
- [5] F. Davoli, A. Giordano, A. Imovilli, C. Nobile, G. Pederiva, S. Zappatore, "AVC R ISA: a MAC layer for NOS/net", *Proc. 9th ARRL Computer Networking Conference*, London, Ontario, Sept. 1990, pp. 16-20.

Spectral Efficiency Considerations for Packet Radio

Phil Karn, KA9Q

ABSTRACT

Radio spectrum is a lot like land; with the possible exception of the Dutch, nobody's making any more of it. So the enormous growth in demand for spectrum means that existing users, especially radio amateurs, either have to find ways to make do with less, be displaced by new users considered more worthy by regulatory agencies, or both.

This paper qualitatively discusses the spectral efficiency of packet radio from several angles, ranging from antenna design, RF modulation and channel access methods to network protocols, routing algorithms and data encoding methods. Maximizing the useful carrying capacity of a spectrum assignment requires a comprehensive look at all of these, factors and more. Digital techniques finally make it possible to exploit these gains, but so far amateur radio has been very slow in adopting them. I hope that this paper will stimulate some work in this direction.

1. Measuring Spectral Efficiency

One seemingly basic problem with spectral efficiency is this: How do you measure it? For example, in the debate over the codeless license, CW was frequently asserted to be more efficient than all other operating modes because it (usually) uses less bandwidth. In effect, the units for measuring spectral efficiency were claimed to be 1/Hz; the narrower the signal, the better.

But this is a naive measure of efficiency because it doesn't take into account the *rate* at which information is being transmitted. A better measure is the ratio of the data rate to the occupied bandwidth; this has units of "bits per second per Hz of bandwidth". Modems are frequently evaluated in this way; indeed, the FCC sets minimum requirements for this figure for commercial digital microwave systems.

Unfortunately, even this figure does not give the complete picture. Very few (if any) individual radio transmitters have exclusive, worldwide use of their channels. Somewhere else that same spectrum is almost certainly in use by other transmitters, and hopefully they *are* all far enough away from each other to avoid mutual interference. This is commonly known as "geographic spectrum reuse". A combination of

directional antennas and/or physical separation (terrain blocking or propagation losses) isolates the transmitters in different geographic areas to avoid harmful interference between them.

So a better measure of spectrum efficiency would have units of "bits per second per Hz of occupied bandwidth per square kilometer". This is the total useful data rate, summed across all of the transmitters in a given geographic area that are sharing a given piece of spectrum.

Yet another factor that we should take into account is the distance involved. Just as the work performed by a *freight* company is the product of cargo weight times the distance it is moved, the measure of useful "work" performed by a data network should be bits times distance. This prevents any apples-and-oranges comparisons of spectral efficiency between wide- and local-area networks. So our final measure of network spectral efficiency has units of "bits per second times distance *moved*, per Hz of occupied bandwidth, per square kilometer of geographic area":

$$E = \frac{rD}{BA}$$

where:

E = network spectrum efficiency figure of merit

r = **total** network traffic capacity in **bits/sec**

D = average distance between source and destination nodes in km

B = total RF bandwidth allocated to the network in Hz

A = geographical **area** occupied by the RF allocation in square **km**

The value E is the figure that we should maximize.

2 Directional Antennas

One effective way to increase **the** carrying capacity of spectrum is by using highly directional antennas, especially on the higher UHF and microwave bands. Glenn **Elmore**, N6GN, has made this point quite eloquently. [Elmore90] I completely agree with Glenn that amateurs should deploy microwave links with directional **antennas** whenever and wherever they are practical. Because microwave antenna patterns can be made so narrow, the design of the associated modem hardly matters (as long as it works). Even with a “low efficiency” modem (in a **bits/sec / Hz** sense), the resulting system spectral efficiency is still far larger than on VHF with omnidirectional antennas because so little geographic area is covered by each microwave beam. **This** allows many different links to share the same spectrum in very close proximity without interference.

However, there are still many situations where point-to-point microwave links are not yet practical, such as portable and mobile stations (particularly in emergency situations), and where line-of-sight microwave propagation is obstructed (e.g., by trees and hills). So VHF and UHF packet radio with conventional low-gain antennas will still be with us for quite some time, and the resulting interference problems must still be dealt with.

3 Interference Limited Systems

This brings up a vitally **important** issue in the calculation of spectral efficiency that is only now getting the attention it deserves. The single **most important factor in the efficient use of spectrum is the minimum geographic spacing required to avoid harmful co-channel interfer-**

ence between transmitters. The closer the allowable co-channel transmitter spacing, the more traffic the channel can support in a given geographic region.

When spectrum is “reused” in this manner, it is **not necessary** that there be no interference (i.e., that any co-channel interference be well below the noise floor of the receivers). It is **only necessary** that a desired signal be **sufficiently** stronger than the undesired signals at a given receiver so that the demodulator can work properly even in the presence of this weak interference. This is **referred** to as operating in an “interference limited” (as opposed to “noise limited”) environment.¹

The interference rejection capability (aka “capture effect”) of the modulation method in use therefore becomes a prime factor in the efficiency calculations. However, low (good) “capture ratios” are **inherently associated** with wide band modulation methods, while narrow band modulation schemes are inherently much more sensitive to interference. Yet low capture ratios are so vital to spectral efficiency that they almost always “pay back” far more than they cost in extra bandwidth by allowing much closer co-channel transmitter spacing. This leads to the somewhat paradoxical fact that by going to a wider (and seemingly less efficient) modulation method, overall spectrum efficiency can often be greatly **increased!** [Costas59] This is why, for example, current cellular telephone systems use FM rather than SSB; FM has a capture effect while SSB does not, so using FM more readily allows the reuse of frequencies in other **nearby** cells. The bottom line is that a FM cellular radio system is more **spectrally** efficient than one using SSB, despite the much wider **FM** channel.² [Lee89] gives this approximate for-

¹ *‘unfortunately, many amateur repeater owners insist on a very wide protection area for their frequency assignment to preclude the accidental triggering of their systems by the weak, distant users of other repeaters sharing the same assignment. They refuse to implement tone-coded squelch (PL) even when it could totally solve the problem because of the advantage given to local users by the FM capture effect. As we will see, this attitude is extremely wasteful of spectrum.*

² It is most unfortunate that the FCC did not understand this paradoxical connection between modulation bandwidth and, spectrum efficiency when they were convinced to reallocate 220-222 MHz to the Land Mobile Service for use with a supposedly more efficient modulation method, SSB.

mula for the frequency reuse factor as a function of the required carrier-to-interference ratio in a hexagonal **cellular radio** system **that uses omnidirectional antennas**:

$$K = \frac{\left[6 \frac{C}{I}\right]^{\frac{2}{\gamma}}}{3}$$

where

K = frequency reuse factor.

$\frac{C}{I}$ = required carrier-to-interference ratio

(expressed as a power ratio, not dB)

γ = propagation slope factor, 2 for free space (inverse square), 4 for a typical terrestrial mobile environment

A given channel can be used in only $1/K$ of the cells. E.g., if $\frac{C}{I}$ is 18 dB (as it is in a FM cellular radio system), then $K=7$ and a given channel can be used in only 1 of every 7 cells. If the required $\frac{C}{I}$ ratio can be decreased to 10 dB, then K would be less than 3; this would more than double the number of transmitters that could share each channel. The next generation of cellular telephony will substantially improve on FM's capacity by going digital, where by the proper choice of modulation method and with forward error correction coding (FEC), the capture ratio can be as low as 7 dB. The most promising scheme, not coincidentally, also has the widest signal bandwidth: CDMA (Code Division Multiple Access, more commonly known as "spread spectrum").³ [Gilhausen91]

³ The big win of spread spectrum for mobile communications is its ability to handle multipath fading. In analog FM, 8 dB of system margin is required to account for fading, over and above the 10 dB C/I ratio required on a nonfading channel. But spread spectrum allows the separation of multipath components, avoiding the rapid "mobile flutter" so characteristic of multipath fading in narrow band FM. Automatic power control easily compensates for the slow propagation variations that remain. This avoids the need for a big fading margin and allows co-channel transmitters to be much more closely spaced. In a network of fixed stations, multipath fading is not as much of a problem, so the important factor in system capacity calculations is just the C/I ratio required by the modulation (and coding method, if any). In both the fixed and mobile cases, the extra bandwidth required by adding FEC usually more than pays for itself in the closer co-channel transmitter spacing it allows.

4. Power Control and Routing

Simply using RF modems capable of good capture ratios isn't enough, **however**; automatic transmitter power control is **also required** to take full advantage of them. With automatic power control, each transmitter ensures, on a continuous basis, that a sufficient signal-to-noise ratio (actually $\frac{E_b}{N_0}$ ratio) **exists** at its intended

receiver **and no** more. Running more power than necessary to yield good performance is like buying higher octane gasoline than your car needs -- it doesn't work any better, and you only squander money and natural resources. The whole purpose of designing modems with good capture ratios is to **allow transmitters** sharing the same channel to be placed more closely together, and this is not possible unless each transmitter uses only the minimum power required to reach any given receiver.

The transmitter power required to reach a given receiver in a network of packet **stations can vary** widely depending on the distance between them, the presence of obstacles, nonideal antenna patterns, etc. So an interesting question **appears**: given the choice of relaying a packet to one of two intermediate stations between the sender and the destination, one of which is close and the other farther away (but closer to the destination), which one should be chosen?

The answer? It depends. If the packet **must** be delivered with the absolute minimum delay, then sending the packet to the relay station that is farther away (and that much closer to the destination) **will** clearly get it there faster, since each relay hop takes time. But the additional transmitter power required to reach the station farther away means that a larger geographical area must be blanketed by the transmission, thus denying the simultaneous reuse of the channel to that many more stations. So if **maximizing** total network capacity is the goal, then the routing algorithm must minimize something other than simply the number of hops taken (or even the total distance traveled) to reach a destination.

Dave Mills, W3HCF has reported on a study of this problem done for the DARPA Multiple Satellite System (MSS). [Mills87] The study concluded that the proper metric to be minimized by an MSS routing algorithm is the sum of the squares of the distances between the nodes (relay satellites). Because of the inverse-square

law in free-space radio propagation, this effectively minimizes the total RF energy, summed over all of the transmitters involved, needed to relay a bit of information to its destination. **This** is true even when more nodes are used than if the routing algorithm simply picked the least number of hops to the destination.

In a terrestrial store-and-forward **network**, propagation losses usually increase with distance much faster than the square of the distance because of obstructions, scattering and **multi**path; the fourth power of the distance is generally used in analysis. But if each node measures the actual **transmitter** power required to **reach** a given destination and reports that as its routing metric for that **link**, then the propagation effects are automatically taken into account when the routing algorithm minimizes the sum of the transmitter powers used in reaching a given **destination**.⁴

5. Channel Access Methods

The algorithms that determine when a station **transmits** are another important factor in a network's overall spectral efficiency. Because of the need to operate in an interference-limited environment, carrier-sense multiple access (CSMA) schemes won't work very well if they always inhibit transmission whenever a signal is heard on channel, no matter how **far** away that other transmitter may be. Such systems are analogous to the repeater operator who refuses to use PL and still complains about remote triggering of his repeater, as mentioned in an earlier footnote. Schemes that rely on receiver feedback (as opposed to channel sensing at the transmitter) to avoid collisions would seem to be the only practical approach to this problem. See [Karn90] for a discussion of one possible approach.

⁴ This tradeoff between delay and network efficiency suggests an interesting use for the long-ignored type-of-service (TOS) bits in the IP header. By default, packets would be routed using the minimum-total-power criteria, but IP datagrams that have the "low delay" bit set would use a different set of routing criteria that minimize delay at the expense of network capacity. This could be quite useful for emergency or priority traffic.

6. Protocols

From the point of view of the packet **subnetwork** designer, the upper layer protocols used **are** irrelevant; they are simply part of the user data that is to **be** moved. However, **from** the user's point of view, everything but his data is overhead. **Therefore** the cost of these protocols could be considered as part of the overall network spectral efficiency equation.

When properly implemented and tuned, the overhead taken by the protocols used in a computer network is usually a second-order **factor** in the overall efficiency of the system. Even the overhead of a "heavy" protocol like **TCP/IP** is easily minimized by using sufficiently large data packet sizes or by compressing headers [Jacobson90]. But this applies only when the protocols are used as intended, e.g., providing reliable point-to-point transfers with TCP. Unfortunately, it is a common practice to use multiple point-to-point protocol connections to emulate a broadcast or multicast **service**; the data is sent N times to N receivers, even when **omnidirectional antennas are** being used and the receivers could easily have shared a common, single transmission. The biggest offender in this regard is the **DX Cluster**, which **in this** author's opinion comes close to being a criminal abuse of the **AX.25** protocol. Another is the common practice of multiple users individually **re-reading** the same public bulletin from a BBS when the bulletin could have been broadcast to everyone at once. Fortunately, protocols designed **specifically** for the efficient broadcast of digital information have been designed and are now being deployed. [Price90] Given that much of the information carried by the amateur packet radio network is of general interest, such protocols should **significantly enhance the effective** capacity of the network. In our network efficiency equation, a broadcast protocol in use by N receivers would effectively multiply **overall** efficiency by approximately N .

7. Compression

Another higher-level issue in spectrum efficiency is the use of data compression. A network doesn't care about the values of the bits it carries; it "costs" just as much to send a million "0" bits as a million-bit text document, even though the useful information content of the former is probably quite a bit less. Users should therefore try to use the network's capacity in the most efficient way possible by compressing their

data before transmission.

Data compression has been well studied and is widely used in the computer field. Public domain and shareware utilities (such as **PKZIP**) are quite common, and they typically yield 50-80% reductions in the size of English text and computer program files. Users can run these utilities manually before sending their files over the network, or they could use the automatic LZW stream compression features built into the NOS TCP/IP package by Anders Klemets. [Klemets91]

Data compression does not increase the capacity of the network per se, it simply uses it more efficiently. But the bottom line is the same: the network can do more useful work (moving user data) with the same spectrum resources.

8. Conclusion

It is the author's belief that an efficient, self-configuring, single-channel half-duplex store-and-forward amateur packet radio network would be quite practical if the design principles discussed here were pursued. The much-maligned "digipeater network" is so bad only because the modulation methods, channel access and routing algorithms are all so sub-optimal, and because there is no power control at all. Properly designed, a collection of "digipeaters done right" would have a lot of practical advantages because of its decentralized nature. All the nodes would be equal, so the failure of any one node need not bring the entire network down, as would happen if the hub or repeater in a centralized network were to fail. (This network model was used for the original DARPA packet radio experiments precisely because of this inherent robustness.) And the system capacity could actually increase as additional nodes were added, because the average inter-nodal distance would decrease, allowing the min-power routing and automatic power control algorithms to reduce average transmitter powers.

As a first step toward such a network, I urge the manufacturers of digital radios to include the "hooks" for automatic power control. What's urgently needed is a way for the packet controller CPU to quickly vary the power of the transmitter in discrete steps, e.g., with a D/A converter, and a way to measure incoming receiver signal levels, e.g., with an A/D converter on the AGC line. Once we have these hardware features, we software types can do the

rest.

9. Acknowledgements

I would like to thank my new colleagues at Qualcomm, Inc, particularly Klein Gilhousen, WT6G and Mike Brock, WB6HHV, for some very enlightening discussions.

10. References

[Costas59] John P. Costas, K2EN, *Poisson, Shannon and the Radio Amateur*, Proceedings of the IRE, December 1959.

[Elmore90] Glenn Elmore, N6GN, *Physical Layer Considerations in Building a High Speed Amateur Radio Network*, ARRL Computer Networking Conference, 1990.

[Gilhousen91] Klein S. Gilhousen, WT6G, et al, *On the Capacity of a Cellular CDMA System*, IEEE Transactions on Vehicular Technology, Vol. 40 No. 2, May 1991.

[Jacobson90] Van Jacobson, *Compressing TCP/IP Headers for Low-Speed Serial Links*, Internet RFC 1144, February 1990.

[Karn90] Phil Karn, KA9Q, *MACA - A New Channel Access Method for Packet Radio*, ARRL Computer Networking Conference, September 1990.

[Klemets91] Anders Klemets, SMORGV, *LZW Compression of Interactive Network Traffic*, ARRL Computer Networking Conference, 1991.

[Lee89] William C. Y. Lee, *Mobile Cellular Telecommunication Systems*, McGraw-Hill, 1989.

[Lee91] William C. Y. Lee, *Overview of Cellular CDMA*, IEEE Transactions on Vehicular Technology, Vol. 40 No. 2, May 1991.

[Mills87] David L. Mills, W3HCF, *Advanced Topics on the DARPA Internet System*, Interop '88 tutorial notes, September 1988.

[Price90] Harold Price, NK6K, and Geoff Ward, K8KA, *Pacsat Broadcast Protocol*, ARRL Computer Networking Conference, September 1990. (See also other related papers in the same issue).

LOSSLESS DATA COMPRESSION ALGORITHMS FOR PACKET RADIO

W. Kinsner, VE4WK

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg, Manitoba, Canada R3T-2N2

Fax: (204) 275-0261

e-Mail: Kinsner@ccm.UManitoba.CA

E-mail: VE4WK@VE4KV.MB.CAN.NA

Abstract

This paper is a review of important data **compression** methods and techniques, showing their evolution **from** the simplest data suppression to the modern adaptive (universal) methods. Selected **lossless** techniques for critically important **data**, requiring **perfect** compression and reconstruction are presented. Lossy techniques for imperfect data such as speech, images, biological signals, and casual text are mentioned.

1. INTRODUCTION

1.1 Motivation

Transmission of data over telephone lines and packet radio, using standard data and file transfer protocols, **as** well as storage of data on magnetic media and in other storage devices can both be improved by data compression techniques. Such techniques reduce space, bandwidth, and input/output load requirements in digital system. For example, the statistical variable-length **Huffman** technique [Huff52] compresses text by 20%. **This** technique requires prior knowledge about the statistics of the file to be compressed **Even better results may be obtained with the** arithmetic coding technique, as implemented by **Witten**, Neal and **Cleary** [WinC87]. The run-length **encoding** used in facsimile can compress a single **81/2x11** inch sheet so that its transmission can be done in 30 seconds on a voice-grade line at 9600 bps. The popular adaptive Lempel-Ziv-Welch (**LZW**) general purpose technique [Welc84] can compress data (text., numeric, mixed, and bitmapped images) by 40 to 60%. This technique does **not** require a **priori** knowledge of the file structure, data types, or usage statistics, **and** can operate on files of any length. **The** compression is noiseless **and** reversible in

that a decompressed file is the exact image of the source file. This contrasts with data reduction and abstraction techniques in which data are deleted from the source. **Nevertheless**, such lossy data compression with **fractals**, neural **networks**, and **wavelets** are important techniques for research and practical applications, as they can achieve impressive compression ratios as high as **10,000:1** [Kins91a].

There are many other techniques capable of compressing and decompressing data efficiently [Stor88], [Held87], [Lync85], [Reg81], [Kins91a]. Which one is suitable for a given data or **file** structure? How should we measure the efficiency of the techniques? How easy is it to implement them? These and other questions have to be answered before using the best methods and techniques. Therefore, the purpose of this paper is to provide a review of the basic data compression methods and techniques, and to show their evolution from the simplest data suppression to the modern adaptive (universal) and lossy methods.

1.2 Models of Data Compression

Data compression refers to the removal of redundancy from a source by a proper mapping into codewords, **carrying all the necessary information about the source so** that &compression could be possible without loss of information. The compression and decompression processes are illustrated in Fig. 1.

A stream of **p** input message symbols (source characters) **M** is compressed into a smaller string of **q** **codewords** $\Delta(M)$ according to a particular algorithm, and passed through a medium (data storage or communication links). At the receiver, the compressed data $A(M)$ are

mapped back into the original source M , without any losses. The compression can be done in either hardware, software, firmware, or any combination of them. Software solutions may be applicable for slow data streams (Mbit/s), while modern parallel pipelined hardware solutions may provide speeds of **hundreds** of Mbit/s.

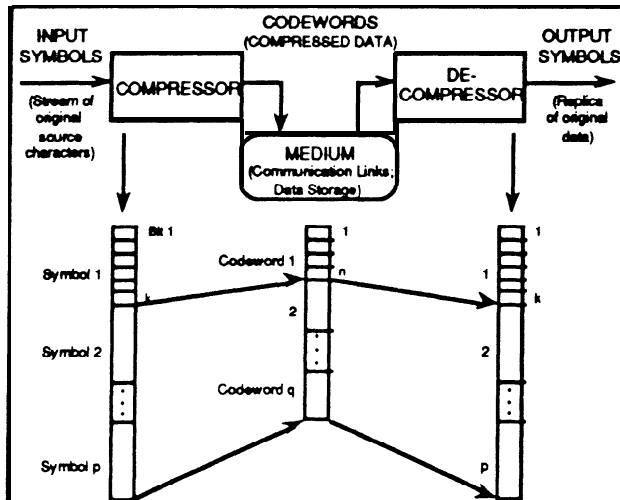


Fig. 1. A model of transparent data compression.

The compression and decompression processes may have a number of attributes, as shown Fig. 2. A compression is reversible if the source data can be reconstructed from the compressed codewords. The compression is noiseless when no information is added into the decompressed data, thus making it the exact replica of the source. It is also **lossless** if no information is removed from the recovered data. For example, the **Huffman**, **LZW** and **WNC** techniques are noiseless and lossless. In contrast, nonreversible (or **lossy**) mapping (data compaction or abstraction) removes **redundancy** using **approximate** methods, and the exact reconstruction of the source is not possible. For example, speech compressed using the linear predictive coding (**LPC**) or adaptive differential pulse code modulation (**ADPCM**) algorithms **cannot** be reconstructed exactly.

Compression is called **transparent** when it is done outside any interaction with a computer programmer. Compression that is not transparent is also called **inferactive**. Compression may be either **statistical** (e.g., **Huffman**) or **nonstatistical** (e.g., **LZW**). In the **statistical** techniques, symbol usage statistics or data types must be **provided in advance, based on** either an average or local analysis of the actual data. Since the **statistics** gathering process requires a single pass and the compression another, these techniques are also **called two-pass techniques**. In contrast, **nonstatistical** techniques employ

ad-hoc rules designed to compress data with some success. The **statistical** techniques may produce codewords that **are** either of **fixed length (LZW)** or **variable length (Huffman)**, with the former giving higher compression ratio.

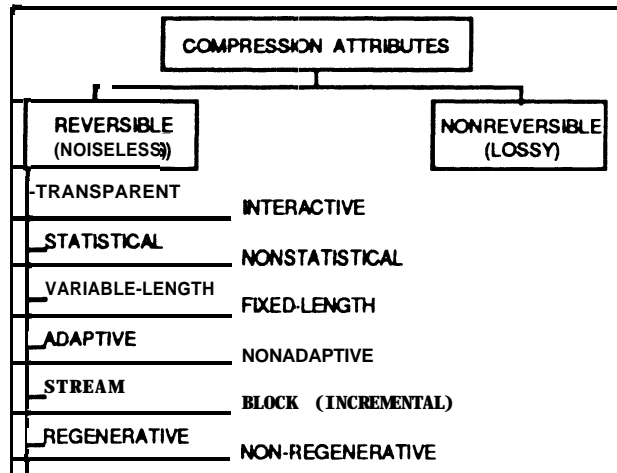


Fig. 2. Attributes of data compression methods.

The statistical and **nonstatistical** techniques may also be classified as either **adaptive** or nonadaptive. The **adaptive** (or **dynamic**) compression does not require advanced knowledge, and develops the necessary translation tables (e.g., **LZW**) or the data statistics (e.g., dynamic **Huffman** and **WNC**) based exclusively on the incoming source stream. They **are** also called one-pass techniques. The **nonadaptive** (or **static**) techniques generate codewords, without affecting the original translation rules or data **statistics**. Depending on the method of outputting the codewords, a compression technique may be classified as **stream** or block. A **stream** technique outputs a codeword as soon as it is available, while the **block** technique must wait until the compression of a block is completed. For example, the arithmetic coding is a block technique, with recent improvements that include **incremental** operation whereby the entire block is broken into smaller **ones** that are output more often (e.g., **WNC**). Compression may also be regenerative or nonregenerative. In the **non-regenerative** techniques, the translation table must be transmitted to the receiver, or else decompression is not possible. The **regenerative** methods do not require the transmission of the translation tables, because they are capable of **reconstructing** the table from the codewords. If the compression and decompression phases take the same effort (time and real estate), then it is called **symmetric**. Clearly, the methods that are reversible, noiseless, lossless, transparent, **adaptive**, regenerative, and symmetric seem to be the most desirable.

1.3 Redundancy and Entropy

The amount of data compression can be measured by the compression ratio defined as

$$R_c = \frac{pk}{qn} \quad (1)$$

where k is the number of bits per symbol in the original message, p is the number of source characters in a message, n is the number of bits in a codeword, and q is the number of codewords. Thus, pk is the number of bits in the original data string, and qn is the number of bits in the compressed string. For example, $R_c = 2:1$ signifies compression by 50%.

Redundancy can also be expressed in terms of entropy of a code alphabet, Γ , corresponding to a string (source) alphabet, S , which in turn is taken from a symbol alphabet Σ . The code alphabet is also called *dictionary*, D , which contains a set of strings, but may also include Σ or even S . The string alphabet is *first-order* if the probability of taking the next symbol does not depend on the previous symbol in the generation process of the string.

Entropy of a first-order source alphabet, S , taken from a binary symbol alphabet $\Sigma\{0,1\}$ (also representing the number of levels in the signal used to transmit the message) can be derived in the following form

$$H_a = -\sum_{i=1}^m p_i \log_2 p_i \quad (2)$$

where p_i is the probability of occurrence of each symbol in S , $m=|S|$ is the number of symbols in the source alphabet, and the base b of the logarithm is equal to the length of the symbol alphabet $b=|\Sigma|$ (2 in this example).

Redundancy R_a in the source alphabet, S , is measured as the difference between a unit of information H_1 for the alphabet, S , and entropy H_a as given by Eq. 2. Thus, if H_1 is

$$H_1 = \log_2 m \quad (3)$$

then

$$R_a = \log_2 m - H_a \quad (4)$$

which indicates that if the character probabilities p_i are equal, the entropy must be $H_a = \log_2 m$, and there is no

redundancy in the source alphabet, $R_a = 0$, implying that a random source cannot be compressed.

The number of bits, λ_i , required to encode a symbol whose probability is p_i can be estimated from

$$\lambda_i = \lceil -\log_2 p_i \rceil \quad (5)$$

where $\lceil x \rceil$ is the ceiling function producing the closest integer greater or equal to x . In practice, the codeword length may not be exactly equal to this estimate. For the actual code, we can calculate the entropy of the code alphabet, Γ , corresponding to the source alphabet, S , by taking the sum of products of the individual codeword probabilities p_i and the actual corresponding codeword lengths, λ_{ci} ,

$$H_c = \sum_{i=1}^m p_i \lambda_{ci} \quad (6)$$

This difference between the code entropy and the source entropy shows the quality of the actual code; if both are equal, then the code is called *perfect in the information-theoretic sense*. For example, Huffman and Shannon-Fano codes are close to perfect in that sense. Clearly, no statistical code will be able to have entropy smaller than the source entropy.

1.4 Classification

Figure 3 shows the major lossless data compression methods, such as the run-length, statistical and adaptive, as well as a class of lossy compression methods used to reduce the bit rates of speech signals, images and biological signals. The programmed and hybrid methods may be either lossless or lossy or combinations of both, and will not be discussed here in detail. This section presents an overview of the major methods, and discusses some techniques belonging to the simple run-length, statistical and adaptive methods.

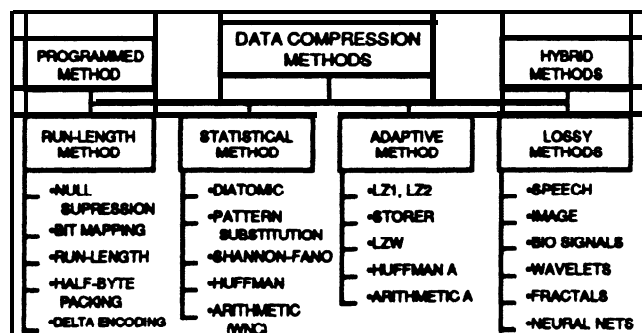


Fig. 3. Compression methods.

2. COMPRESSION METHODS

2.1 The Programmed Method

The programmed method requires interaction with the application **programmer**, who determines an efficient way of removing application-specific redundancy from the source, and programs it into the application. Such software is not very portable, and the development cost may be prohibitive. So, this nontransparent approach will **not** be discussed here. Instead, we shall concentrate on methods that have universal techniques and **corresponding** algorithms that can be made transparent

Another form of a programmed data compression is an object-oriented description of an image, and a programmed description of attributes. For example, Adobe's PostScript may reduce the facsimile image by 10: 1 and considerably reduce the description of fonts transmitted for printing.

2.2 The Run-Length Method

As shown in Fig. 3, this method includes the following techniques: null suppression with a pair, null suppression with a bit map, arbitrary character suppression with a triple, nibble (half-byte) packing, and delta encoding. These techniques will be described briefly next.

An early compression technique, the **null suppression with a pair**, was designed to suppress blanks (nulls) in the IBM 3780 bisync protocol, with **30-50%** throughput gain. A sequence of blanks is substituted by an ordered pair of characters (**S_c**, Count) where **S_c** is a special suppression indicator, and is followed by the number of blanks to be suppressed. For example, a string **ABCbbbbbbXYZ** is reduced to **ABCS_c7XYZ**.

Another technique is **the null suppression with a bit map**. If the specific characters such as blanks are distributed randomly within a string, the blank suppression technique cannot be used. Instead, we can subdivide the string into substrings whose length matches the number of bits in each character (word), and the **position** of the **blanks** in a substring can be marked with individual bits in **the** word, thus creating a bit map **whose** compact representation can be placed within each substring.

This general run-length technique may be modified for facsimile which has many white (0) and **black** (1) pixels. **The pixels may be** represented by individual bits in a word in a bit-mapped manner. **Thus**, four bytes **can** carry 32

pixels. If we encode a group of white pixels by a single byte and **black** pixels by another byte, then the run-length compression can be improved. **For** example, a four-byte sequence containing the numbers **150/10/220/5** represents 150 white pixels, followed by 10 black pixels, which **are** followed by 220 white and 5 black pixels. This has the potential of **32:1** compression ratio (total 1024 pixels against 32 when each bit represents a single pixel only).

Another technique is the **arbitrary character suppression with a triple** which is a **direct** extension of the null suppression to an arbitrary character. Instead of the ordered pair **characters**, we must now use three characters: the special **suppression** indicator, followed by **the repeated character and the character count**.

The nibble (half-byte) packing is a modification of the run-length and bit-mapping **procedures**. Rather than suppressing the repetitive characters, different characters may have identical upper nibbles which could be suppressed. For example, the upper nibble in ASCII financial characters could **be** suppressed, and the lower nibbles could be packed. The same would apply to the EBCDIC **representation**.

Still another technique, the **delta encoding**, is an important extension of nibble packing. This relative encoding scheme applies to substrings that are similar, thus the difference (delta) between the substrings would contain only few non-zero bits or characters, and such delta substrings could be compressed using the above suppression techniques. Examples of such strings could be found in telemetry and facsimile.

In telemetry data, if measurement data are slowly varying within a period of time, their values do not change significantly and only differences could be transmitted, leading to smaller numbers which could have more compact representation. The delta encoding could run until either the range **of** the small numbers is exceeded or data integrity would **require** a periodical reset of the absolute value.

In facsimile data, the straightforward run-length encoding can be used to suppress the white or black pixels. The delta technique further reduces the **scan** line **data**. If a reference line has been transmitted without compression, only the difference of the subsequent lines **may be required to reconstruct the source data**.

2.3 The Statistical Method

This method includes the following techniques:

diatomic encoding, pattern substitution, and **variable-length** encoding.

The *diatomic encoding* technique takes two **characters** and **replaces** them with a single character, thus having **fixed** length and a possible 2:1 compression ratio. Since **the number of special characters is limited, not all the pairs can be compressed. Instead, the most frequent pairs are selected for compression.** The pairs must first be **identified** by a file analysis **program**, and the **most frequent pairs** are selected as candidates for **compression.** Extensive studies have been done on different data types to identify such pairs for English, text, FORTRAN, COBOL and BASIC [e.g., Held87].

The *pattern substitution* technique is an extension of the **diatomic encoding** in that it also assigns special short codes to common multicharacter groups. This scheme may benefit files containing computer languages (e.g., WRITE, READ, etc.) or natural languages (“and”, “the”, etc.). The substitution can be done by single codes or special pairs such as \$n, otherwise never appearing in the string. Compressions of 20% were reported.

Another technique is *the variable-length encoding.* All the previously discussed techniques employ **fixed-size** character **codes** and **produce fixed-size** codewords. A more compact code can be achieved by assigning shorter codewords to **frequent** characters and longer codewords to less frequent characters or their groups. This approach was used by Samuel Morse when he selected short sounds for the **frequent** characters (E,T,A,N) and longer sounds for the less frequent ones. There are three effective variable-length **techniques** such as the ordinary and **generalized** step **codes**, the Shannon-Fano, **Huffman**, and WNC arithmetic coding, as described in Section 3.

2.4 The Adaptive (Universal) Method

Adaptive (or universal) techniques can be seen as **generalizations** of many of the **previous** techniques. The key idea here is that **no statistical knowledge** is necessary to convert the **available** input stream into codewords and vice versa. Instead, an optimal conversion is possible, **using** a form of *learning* about *the structure* of either the source stream or the codeword stream. Thus, the **techniques are capable of producing good codewords, either** without prior knowledge about the data statistics, or even better codewords with data statistics acquired by observation during the compression phase. **Examples of such adaptive statistical nonregenerative techniques include the Huffman and arithmetic code WNC. Examples of adaptive nonstatistical regenerative techniques are** due to Lempel and Ziv (LZ1 [LeZi76], [ZiLe77] and LZ2

[ZiLe78]), Storer (heuristic dictionary algorithms) [Stor88], and Lempel, Ziv and Welch (LZW) [Welc84], [KiGr91]. **Other adaptive methods are being developed, including an algorithm for binary sources and binary code alphabet, a memory efficient technique based on fused trees, techniques requiring a small number of operations per encoded symbol. Some of these techniques are described by Kinsner [Kins91a], and the basic ideas behind the LZW technique are presented in Section 3.**

2.5 Lossy (Approximate) Methods

The previous methods and techniques are **lossless and noiseless** in that nothing is lost during the data compression and nothing is added during the reconstruction **processes, respectively. These properties are essential in perfect data in which a loss of a single bit may be catastrophic [Kins90].** On the other hand, imperfect data such as speech, images, biological signals, and casual electronic mail text may tolerate minor changes in the source and still be sufficient after their reconstruction, according to a distortion measure. This class of lossy techniques may produce extremely large compression ratios (10,000: 1). There are **three** emerging classes of compression techniques with enormous potential based on **fractals**, neural networks, and wavelets. We are working on all **three** techniques related to speech and pictures [Kins91a], [Kins91b].

3. EXAMPLES OF ALGORITHMS

3.1 The Shannon-Fano Coding

The **Shannon-Fano** (S-F) code has efficiency approaching the theoretical limit and is the shortest average code of all statistical techniques. The code is also good because it has the self-separating property (or the **prefix property**) because no codeword already defined can become a **prefix** in any new **codeword.** This property in the S-F code leads to the ability to decode it “on the fly”, without waiting for a block of the code to be read first. **The main question is how to generate such an optimal, information efficient and self-separating code? The answer is in the application of an old principle of binary search tree, BST (or halving, or binary chopping). How should we halve the entire set of symbols to achieve the optimum variable-length code? A viable approach is to use the overall (joint) probability of symbols and assign the first 0 (or 1 –the choice is arbitrary at this first step) to the symbols above 0.5 and the first 1 (or 0) to those with joint probability not greater than 0.5. This subdivision continues on the halves, quarters and so on, until all the symbols are reached, as shown in Fig. 4.**

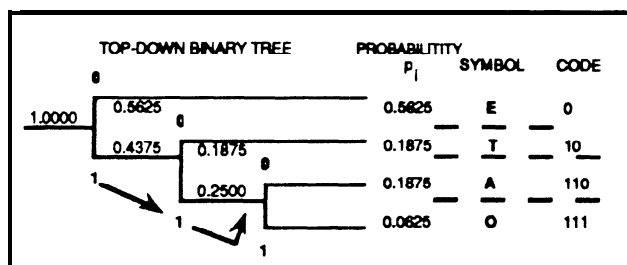


Fig. 4. Shannon-Fano top-down binary tree.

3.2 The Huffman Coding

The **Huffman** coding scheme [Huff52] belongs to the same class as the Shannon-Fano technique in that **both** are statistical, variable-length, optimal-length, self-separating, and use the binary decision principle to create the code. The difference lies in the application of the BST; i.e., the S-F code is created by top-down binary chopping, while **the Huffman code** is formed by bottom-up **binary fusing**. As already described, the S-F code is created by subdividing the symbol table set into two groups whose joint probability is as close to 0.5 as possible. Then, the halves are divided into quarters, and so on, until all the individual symbols are covered. The method is termed top-down because the process starts from the root of the tree where the global probability is 1.0 and progresses to the leaves.

In contrast, the **Huffman** code formation starts from the leaf level and progresses to the root of the tree by fusing the probabilities of the individual leaves and branches, as shown in Fig. 5. Thus, this method emphasizes the small differences between the leaves, while the S-F technique operates on averages. Although the two techniques appear to be the same on short codes, the differences will be discussed later in the section.

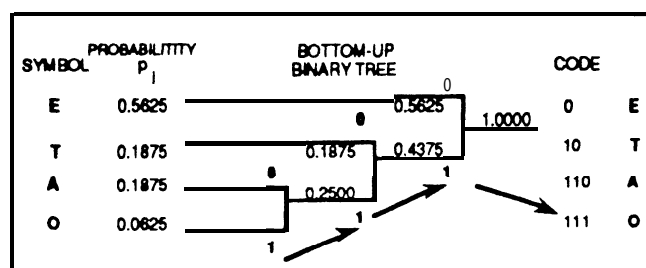


Fig. 5. **Huffman** bottom-up binary tree.

We have implemented both S-F and **Huffman** compression techniques, using **the** C language on the IBM PC. The code is described in [DuKi91a], and experimental results are **presented** in [DuKi91b, this conference].

3.3 Dynamic and Higher-Order Huffman

The S-F is better than the **Huffman** code on alphabets with **spread** probabilities (large variance), while the **Huffman** is better on alphabets with an even distribution of the probabilities (small variance). Although very attractive **from** the **efficiency** point of view, both methods have problems. The problems include the limited size of the translation table (a **256-entry** table for **8-bit** symbols is sufficient for single **characters** only), **difficult decoding** process on binary **trees**, **the a priori** knowledge of the alphabet statistics, leading to two passes over the data to be compressed, and variability in the statistics on large files requiring an adaptive analysis of the data and transmission of **more** than one translation table.

There are many modifications of the basic static **Huffman** and S-F algorithms to cope with those and other problems [Welc84, Regh81, Held87, Stor88]. In the original paper [Huff52], **Huffman** considered non-binary symbol alphabets, $|S| > 2$ in addition to the binary case. Others **considered** code alphabet Γ with unequal cost, as well as source alphabet **with** inaccurate probabilities and **trie** (to distinguish it from a tree, [Stor88]) construction. Implementation issues are discussed in [Pech82] and [McPe85]. Many other references are provided by Storer [Stor88, Chapter 2] and [Kins91a].

The **dynamic Huffman** code was introduced by Faller [Fall73] and Gallager [Gall78]. A naive dynamic **Huffman** encoding and decoding would start with a source alphabet, S , whose symbols have equal probability. The probability distribution is then updated after a character has been encoded or decoded. The **trie** (not tree) construction algorithm must be deterministic for both the encoder and decoder so that the key procedures such as halving decisions and tie breaking are done identically. Although optimal, this method is inefficient because the **trie** must be reconstructed after each character in the message stream. Improvements and generalization of the dynamic algorithm, as well **as** an extension of the **balanced trie concept** is the splay tree and an alternative to this approach the transposition heuristic, **are** discussed in [Kins91a].

3.4 Arithmetic Coding

Arithmetic coding belongs to the statistical **method**, but is different from the **Huffman** class of coding techniques in that it develops a single codeword **from** the input symbol stream by interval scaling, using the **fractal** similarity principle. (Notice that this **fractal** similarity principle has never been pointed out in the arithmetic coding literature before). In contrast, the **Huffman**

technique develops codewords by a table lookup procedure, with the table obtained from a bottom-up BST. Both techniques use statistical models of data that can be either obtained *a priori* (fused model) or adaptively (dynamic model). While Huffman coding generates codewords in response to individual symbols (stream technique), arithmetic coding either waits for all the symbols to arrive prior to sending the codeword (block technique) or outputs partial codewords when the interval has been located up to a predefined threshold (incremental technique). Arithmetic coding, and particularly the implementation by Witten, Neal and Cleary [WiNC87] is shown to be superior to Huffman coding. A tutorial on arithmetic coding is provided by Langdon [Lang84], [Kins91a], and a class of such techniques is presented by Rissanen and Langdon [RiLa79]. We are working on an implementation of the arithmetic coding.

3.5 Lempel-Ziv-Welch (LZW) Coding

The well known static LZ1 [LeZi76], [ZiLe77] and dynamic LZ2 ([ZiLe78]) nonstatistical algorithms due to Lempel and Ziv can be classified as adaptive (universal) techniques. The key i&a here is that *no statistical knowledge* is necessary to convert the input stream into codewords and vice versa. Instead, an optimal conversion is possible using a form of *learning* about the structure of either the source stream or the codeword stream, thus justifying the name “adaptive”. This class of techniques converts variable-length input strings to constant-length output codewords, using a dictionary (also called a conversion table). In the case of Huffman-type coding, the input strings are of constant length, while the output codewords are of variable length, and a *fixed-length* dictionary is provided to both the encoder and decoder. In the case of Lempel-Ziv-type coding, the dictionary is of *variable length*, and the encoder creates its local dictionary, D , in step with the incoming strings. Similarly, the decoder reconstructs D in step with the incoming codewords, thus making the method *regenerative*.

The learning in the algorithm depends on the following four heuristics: (i) initialization heuristic, *IH*, (ii) matching heuristic, *MH*, (iii) updating heuristic, *UH*, and (iv) deletion heuristic, *DH* [Stor88], [Kins91a]. The dictionary D is first *initialized to an* initial string set D_0 which includes at least the string alphabet \mathcal{S} in Storer’s dictionary approach, or is empty in the Lempel-Ziv approach. The encoder dictionary D is constructed by repeatedly matching the incoming character stream to the entries in its D , until a new string s_m is found (or until any other significant event occurs). The compression

results from a substitution of s_m with an index representing the string (provided the length of the index is smaller than $|s_m|$ where $| \cdot |$ signifies the length of an object (\bullet)). Now, D is *updated* according to the matching s_m and the current contents of D . If the new update exceeds the size of D , then some entries must be *deleted*, using a deletion strategy. Since none of the four major activities has a single form, they are called heuristics. It is seen that a large number of adaptive techniques can be derived, depending on the choice of the heuristics.

The LZ1 adaptive (universal) technique has received extensive attention in literature because it can be the perfect algorithm in the *information-theoretic* sense and may be the best algorithm for many applications. Serial and parallel implementations of the scheme are discussed in [Stor88]. The LZW algorithm is another implementation of the LZ1 algorithm [Welc84]. Our LZW code is based on Nelson’s and Regan’s implementations [Nels89], [Rega90], and is included in [DuKi91a]. This C language implementation runs on an IBM PC, and is portable to machines supporting 16-bit integers and 32-bit longs. It is limited to arrays smaller than 64 Kbytes on the MS-DOS machines (12 to 14 bits used). Experimental results obtained with our benchmarks are presented in [DuKi91b, this conference].

4. STATISTICAL ANALYSIS OF FILES PROGRAM

In addition to implementing the Shannon-Fano, Huffman and LZW algorithms, we have also developed a statistical analysis of files (STAF) program for data compression [DuKi91a]. The program is required by all the statistical techniques to design optimal codes for the data streams to be compressed. It first gathers all the vital statistics about the specific file and then estimates the best run-length or statistical technique for the file. Finally, it reports on the findings, and develops an optimal Shannon-Fano and Huffman codes automatically. Notice that the program is not required by nonstatistical techniques such as the LZW.

5. OTHER IMPLEMENTATIONS

The popular adaptive LZW algorithm has very simple logic, leading to inexpensive and fast implementations. Good LZW implementations use 9- to 16-bit codes, handling most applications. A 12-bit code is suitable for medium-size files. Efficiency improves with larger codes. A tight coding of the algorithm can compress 75 Kbytes

in a second on a 1 MIPS machine. The LZW technique can be found in several file compression utilities for archival storage.

For example, the **MicroSoft** MS DOS environment has enjoyed archival programs such as ARC by System Enhancement Associates of Wayne, NJ, as well as **PKZIP** by PKWARE of Glendale, WI. The ARC program has been **ported** to Unix, **CP/M** and other **operating** system environments. Machines with Unix can use the **COMPRESS** and **COMPACT** utilities. The Berkeley Unix has a **COMPRESS** command which is an implementation of the **LZ** algorithm, with a table of up to **64 K** entries of at least **24** bits each (total of over **1,572 kbits** or over 1% kbytes on most machines). The Apple Macintosh environment has several good programs, including **PackIt** IX/III and **UnpIt** by Harry Chelsey [Ches84], as well as **StuffIt** and **UnStuffIt** by Ray Lau [Lau87]. The **StuffIt** shareware is written in Lightspeed C. An improved version of **StuffIt** is now distributed as a commercial package [Stuf90]. The ARC 5.12 program uses a simple RLE and **LZW** algorithms for compression. The **PackIt** program uses **Huffman** encoding only. In the **StuffIt**, compression is done by the **LZW** and/or **Huffman** algorithms, and when they fail, by the less efficient RLE algorithm. Its LZW implementation is similar to the ARC 5.12 **LZW**, but uses 14 bits with a hashing table of size 18,013, rather than the 12/13 bits used in ARC. The ARC **LZW** implementation, in turn, is similar to that of the public domain **COMPRESS** utility in Unix. A recent **LZW** implementation for packet radio by Anders Klemets [Klem90] is designed to work in conjunction with the IBM PC implementation of the **TCP/IP** protocols by Phil Kam of **Bellcore**.

In addition, the **LZW** algorithm can be employed not only on files requiring perfect transmission (e.g., financial data), but also on imperfect data such as e-mail text of non-critical nature, weather data, and digitized speech transmitted using the store-and-forward mode. Files with **poor** data structures (sparse encoding of data and empty spaces) can **also** benefit from **LZW** compression.

6. CONCLUSIONS

This paper presents a classification of the major data compression **methods** and a number of useful compression techniques that could be suitable for packet radio. Although the top-down **Shannon-Fano** technique is better than **Huffman** on alphabets with large variance, while the bottom-up **Huffman** technique is better on uniform alphabets, the latter may employ heuristics to make it better on **all alphabets**. The arithmetic **coding** technique is

better than **Huffman**. The **Storer** static and dynamic sliding **dictionary techniques** are implementations of the **LZ1** and **LZ2** algorithms, with essential generalizations to heuristic algorithms. The popular Lzw technique is **also** an implementation of the **LZ** algorithm.

In addition to the **lossless** techniques, lossy algorithms for compression of imperfect data such as noncritical electronic mail text, images, speech and other biological data, should also be **considered** in packet radio.

ACKNOWLEDGEMENTS

This work was supported in part by the University of Manitoba, as well as the **Natural Sciences and Engineering Research Council (NSERC)** of Canada.

REFERENCES

- [Ches84] H. Chesley, "**PackIt**." (Address: 1850 Union St. #360; San Francisco, CA 94123.)
- [DuKi91a] D. Dueck and W. Kinsner, "A program for statistical analysis of files," *Technical Report, DEL91-8*, Aug. 1991, 50 pp.
- [DuKi91b] D. Dueck and W. Kinsner, "Experimental study of Shannon-Fano, **Huffman**, **Lempel-Ziv-Welch** and other **lossless** algorithms," *Proc. 10th Computer Networking Conf.*, (San Jose, CA; Sept.-29-30, 1991), this *Proceedings*, 1991.
- [Fall73] N. Faller, "An **adaptive** system for data **compression**," *Proc. Seventh IEEE Asilomar Conf. Circuits & Systems*, pp. 593-597, Nov. 1973.
- [Gall78] R.G. Gallager, "Variations on a theme by **Huffman**," *IEEE Trans. Information Theory*, vol. *IT-24*, pp. 668-674, June 1978.
- [Held87] G. Held, *Data Compression: Techniques and Applications, Hardware and Software Considerations*. New York (NY): Wiley, 1987 (2nd ed.), 206 pp. (QA76.9.D33H44 1987)
- [Huff52] D.A. **Huffman**, "A method for the construction of **minimum-redundancy** codes," *Proc. IRE*, vol. 40, pp. 1098-1101, Sept. 1952.
- [KiGr91] W. Kinsner and R.H. Greenfield, "The **Lempel-Ziv-Welch (LZW)** data compression algorithm for packet **radio**," *Proc. IEEE Conf. Computer, Power, and Communications System*, (Regina, SK; May. 29-30, 1991), 225-229 pp., 1991.
- [Kins90] W. Kinsner, "Forward **error** correction for imperfect data in packet radio," *Proc. 9th Computer Networking Conf.*, (London, ON; Sept. 22, 1990), pp. 141-149, 1990.

- [Kins91a] W. Kinsner, "Review of data compression methods, including Shannon-Fano, Huffman, arithmetic, Storer, Lempel-Ziv-Welch, fractal, neural network, and wavelet algorithms," *Technical Report, DEL91-1*, Jan. 1991, 157 pp.
- [Kins91b] W. Kinsner, "Lossless and lossy data compression including fractals and neural networks," *Proc. Int. Conf. Computer, Electronics, Communication, Control*, (Calgary, AB; Apr. 8-10, 1991), 130-137 pp., 1991.
- [Klem90] A. Klemets, "LZW implementation for KA9Q Internet Package NOS," *Packet Radio Bulletin*, 5 Oct. 1990.
(The source of the code is available from sics.se under filename archive/packet/ka9q/nos/lzw.arc, and the NOS is available from thumper.bellcore.com.
Address: Anders Klemets, SMORGV, Sikv 51, S-13541, Tyreso, Sweden.)
- [Lang84] G.G. Langdon, "An introduction to arithmetic coding," *IBM J. Res. Dev.*, vol. 28, pp. 135-149, March 1984.
- [Lau87] R. Lau, "StuffIt." (Address: 100-04 70 Ave.; Forest Hills, NY 1137505133.)
- [LeZi76] A. Lempel and J. Ziv, "On the complexity of finite sequences," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 75-81, Jan. 1976.
- [Lync85] T.J. Lynch, *Data Compression: Techniques and Applications*. Belmont, CA: Lifetime Learning, 1985, 345 pp. (ISBN 0-534-03418-7)
- [McPe85] D.R. McIntyre and M.A. Pechura, "Data compression using static Huffman code-decode tables," *J. ACM*, vol. 28, pp. 612-616, June 1985.
- [Nels89] M.R. Nelson, "LZW data compression," *Dr. Dobbs's J.*, vol. 17, pp. 29-36, Oct. 1989.
- [Pech82] M. Pechura, "File archival techniques using data compression," *Comm. ACM*, vol. 25, pp. 605-609, Sept. 1982.
- [Rega90] S.M. Regan, "LZW revisited," *Dr. Dobbs's J.*, vol. 18, pp. 126-127 and 167, Jun. 1990.
- [Regh81] ELK. Reghbati, "An overview of data compression techniques," *IEEE Computer*, vol. 14, pp. 71-75, Apr. 1981.
- [RiLa79] J. Rissanen and G.G. Langdon, Jr., "Arithmetic coding," *IBM J. Res. Dev.*, vol. 23, pp. 149-162, March 1979.
- [Stor88] J.A. Storer, *Data Compression: Method and Theory*. New York (NY): Computer Science Press/W.H. Freeman, 1988, 413 pp. (QA76.9.D33S761988)
- [Stuf90] "StuffIt Deluxe," *MacUser Magazine*, vol. 28, pp. 68-70, Dec. 1990. (Address: Aladdin systems, Deer Park Center, Suite 23A-171, Aptos, CA 95003; Tel. 408-685-9175)
- [Welc84] T.A. Welch, "A technique for high-performance data compression," *IEEE Computer*, vol. 17, pp. 8-19, June 1984.
- [WiNC87] I.H. Witten, R.M. Neal, and J.G. Cleary, "Arithmetic coding for data compression," *Comm. ACM*, vol. 30, pp. 520-540, June 1987.
- [ZiLe77] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Information Theory*, vol. IT-23, pp. 337-443, May 1977.
- [ZiLe78] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Information Theory*, vol. IT-24, pp. 530-536, Sept. 1978.

LZW Compression of interactive network traffic

Anders Klemets, SMØRGV

Swedish Institute of Computer Science
Box 1263
s-164 28 Kista
SWEDEN

ABSTRACT

This paper summarizes some aspects of data compression and describes how it can be used to increase the throughput of a computer communications link. The popular LZW algorithm is studied in detail, and special consideration is given to the problems that arise when using LZW to compress interactive traffic. An LZW implementation for the KA9Q Internet package for compression of interactive data is presented. Finally, the algorithm used in this implementation is compared to the V.42bis modem data compression standard

1. Introduction

There are different ways to increase the effective transfer speed, or throughput, of a computer communications link. An obvious way might be to increase the bit rate of the hardware. This will usually involve obtaining faster modems, or switching from an RS-232 interface to some other faster interface. But there might be factors that limit the extent to which the physical speed can be increased. For instance, all physical media have a limited bandwidth. For telephone lines the bandwidth is only about 3 kHz. For radio channels the bandwidth is primarily limited by regulations. The usefulness of the equipment may also be limited by its price and by the need for backwards compatibility.

In some cases one may achieve substantial improvement of throughput by replacing or modifying the different communications protocols that are in use. If the link level protocol has not been properly designed, it might impose considerable overhead. It has also been shown to be possible to make significant improvements to the performance of network and transport protocols. In some cases just altering the timing and flow control mechanisms is enough [Jacobs88].

It might actually be possible to improve the transfer rate by adding yet another protocol to the existing protocol stack. This paper describes how a data compression protocol can be used at the presentation layer.

2. Compression methods

Compression methods can be grouped into different categories. For instance, when compressing sound or images, it can make sense to discard some of the information. The decompressed data will not be identical to the original, but when done correctly it may not be noticeable to humans. Such compression is called irreversible. In the general case, however, one would like to have reversible compression, i.e. the decompressed data should be identical to the original data. Anything less would be unacceptable when, for example, transferring binary computer programs.

There are two important compression techniques: statistical coding, and dictionary coding. In statistical coding; each character is assigned a codeword based upon how frequently it occurs. Common characters get short codewords and the more unusual characters get longer codewords. In dictionary coding, strings of characters are entered in a dictionary. Whenever the string occurs, a codeword representing

its dictionary entry is used. See a compression textbook *such* as [Storer88] for a more comprehensive description of these techniques.

Dictionary *coding* is always less efficient than statistical coding. [Bell89] (It gives a lower compression ratio. Compression ratio is defined as the size of input data divided by the size of its compressed equivalent.) But dictionary coding is usually faster than statistical coding. This is especially important when choosing an algorithm for compressing data in real time. The compression algorithm must be able to generate compressed data at the speed supported by the underlying network layers divided by the compression ratio. Similarly, the decompression routine must be able to generate decompressed data at the speed of incoming compressed data divided by the compression ratio. If any of these two relations do not hold, the compression method will not increase the throughput of the communications link. However, in this case it might still be worthwhile to use compression. For instance, there might be a cost associated with the amount of data transmitted over the connection. Compressing the data would lower this cost. On a slow shared channel (e.g. a low speed packet radio channel) compression might be beneficial to the **overall** usage of the channel, even if the compression does not increase the individual throughput.

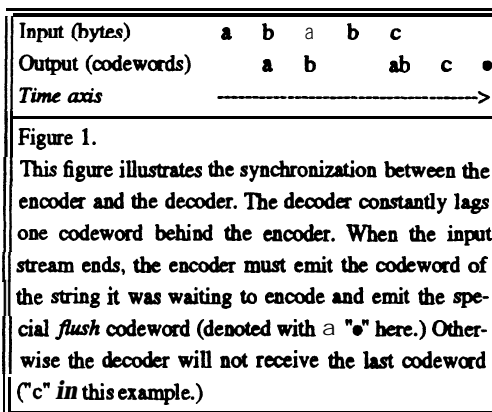
3. LZW compression

Ziv and Lempel [Ziv77, Ziv88] presented efficient compression methods that use dictionary coding. Several variants of **the Ziv-Lempel** methods exist, the most widely used seem to be the LZW method [Welch84] and its derivatives. A pleasing characteristic of these methods is that the encoder and decoder **will** build their respective dictionaries dynamically. The encoded data itself is used to build the dictionaries. At no point does the content of the dictionary have to be transferred.

An LZW encoder that operates on sequences of 8 bit bytes **will** emit codewords that are of a fixed width (at least 9 bits.) The dictionary is initialized to contain codewords representing single bytes with the values 0-255. The operation of the encoder is to read bytes from the input **stream**, concatenating them into a string. It **proceeds** with reading bytes as long as it can find an entry for the string in its dictionary. If the string is not in the dictionary, it will be added, and a new codeword will be assigned

to the new dictionary entry. Following this, the codeword **for** the previous **version** of the string will be emitted. **The** only difference between the previous and the current version of the **string is the last byte. This last byte will now be used as the first and only byte in the string. The** process is then **repeated** by reading yet another byte **from** the input stream.

The following example is in accordance with the original **LZW** specification, with the addition of a special *flush* axleword. Its use is described later. Let us consider the encoding of the string "ababc." The encoder will read the character 'a' from its input stream. **The** string is now "a", and since it is a one byte string, we know that it is always in the dictionary. Then 'b' is read. The string, "ab", is not found so it is added to the dictionary and the codeword for "a" is emitted. **The** string now becomes "b". The character 'a' is read. The string "ba" is not found, and is thus added to the dictionary. Then the codeword for "b" is emitted. The string is now "a" and 'b' is now read. **The** string "ab" is found in the dictionary, so yet another character, 'c' is read. "abc" is not in the dictionary, so it is added, and the codeword for "ab" is emitted, and the string becomes "c". At this point there are no more characters in the input stream. We are supposed to hold on to the string "c" at least until we have read one more character, but since the input stream is empty, we will simply transmit the codeword for "c" followed by a **special flush** codeword. We will refer to this from now on as the "flush" operation. The *flush* codeword is needed to indicate that the normal flow of compression has been broken by flushing the content of the string onto the output stream.



The decoder will build its dictionary only using the codewords it is receiving. When

receiving a codeword, it will take the **first** character in the **string** that the codeword represents, and concatenate it with the string represented by the previously received codeword. The resulting **string** will be added to the dictionary.

In the example above, the codeword for "a" is received. There was no previously received codeword, so no particular action is taken with respect to the dictionary. The codeword is now decoded into 'a' which is emitted to the decoded output stream. Then the codeword for "b" is received. The concatenation of the "a" and the first character from the string "b" gives the string "ab". It is added to the dictionary and 'b' is sent to the output stream. Then the codeword for "ab" is received, causing the string "ba" to be added to the dictionary. Characters 'a' and 'b' are sent to the output. The codeword for "c" is received. The string "abc" is added to the dictionary, and 'c' is sent to the output. Now the special *flush* codeword is received. It causes the recording of the last received codeword to be cleared. Without it, should compression later resume, the decoder would construct a string **beginning** with "c" from the first codeword. That string would then incorrectly be added to the dictionary, creating havoc. Incidentally, since the decoder constructs new codewords using previously received codewords, the slightest data transmission error may cause large parts of the dictionary to become bogus.

It may sometimes happen that the decoder receives a codeword that it does not yet have in its dictionary. Consider for instance if the input stream consists of the string "ccccc". This will be encoded as the codewords for "c", "cc", and "cc" followed the flush codeword. When the first "cc" codeword is received, it will be unknown by the decoder. However, this situation can only arise if the encoded string is the same as given by the previous codeword followed by the first character in that same string. Evaluating this rule **will** yield the desired string, "cc".

4. Compression of transport protocol data

There has been recent studies [Jacobs90] on how to compress the headers of the TCP/IP protocol. This improves the efficiency of TCP/IP when sending very **small** packets, such as single characters typed at a keyboard. For larger size packets, the importance of TCP/IP header compression diminishes, as the importance of compression of the actual data that is sent by

the transport protocol (e.g. TCP) increases.

Sometimes the transport protocol is only used to transfer a **file**. (Consider for instance the data TCP connection during an FTP file transfer.) In that case, almost any data compression method that can operate on the data "on the fly" **will** work. **All** variants of Ziv-Lempel compression **will** work in this situation. When the end of the **file** is reached, the encoder **will** perform the "flush" operation at least partially. It will encode the character string that it is currently holding and emit **its** codeword to the output. However, it is not necessary to emit any **special flush** codeword, **since this was the** last compressed data that was sent on the transport connection. The transport connection is supposedly terminated shortly afterwards, so a special *flush* codeword would serve no purpose here.

Another situation that resembles a file transfer, is the transfer of electronic mail (an SMTP session for instance.) However, here there are interactive elements. In the case of SMTP, for example, there is an initial dialogue between the sender and receiver of the message. When LZW or a similar method is used to compress the data, the decoder will always lag one codeword behind the encoder. This makes it necessary to perform the flush operation after each SMTP command and after each mail message. Consider what would otherwise happen. Most SMTP mail transfer agents will operate in stop-and-wait mode. They will send a command and they **will** not proceed until a response has been received. Using standard LZW (non-flushing) compression, however, the SMTP server will not receive the complete command because its LZW decoder has not received the last codeword of the compressed command. The SMTP server will not do anything until it has received a complete command, but the mail transfer agent will not do anything either, because it believes that its **latest** command has been fully transmitted. A deadlock has occurred.

The same problem **will** arise when transmitting any kind of interactive data. An example application would be characters or lines of characters transmitted in transport protocol packets. (E.g. Telnet running in remote echo and local echo mode, respectively.) Even if there may not always be a deadlock, the lag of the LZW decoder is likely to cause some inconvenient effects. For instance, when using remote

echo, the echoed characters might not be received until several other characters have been typed.

A reasonable general solution to avoid the problem and possible deadlock described above, would be to perform the flush operation whenever the application program passes data for transmission to the layer below it. In order to keep software modules layered, it would be considered a good thing if the application program was as independent of compression as possible, and vice versa. Unless otherwise informed, the compression layer should assume that any data buffers are for immediate processing by the remote peer. Every buffer sent to the transport level would be “flushed.” This would not be necessary when transferring files, or when doing other non-interactive operations, but the overhead would only be two codewords per data buffer. (The codeword of the last string, and the actual *flush* codeword) It would be possible to have the application program explicitly tell the encoder when to flush, but that could introduce complicated **interactions** between protocol layers.

Incidentally, suppose that Telnet is being run in remote echo mode. To echo each character after it has been typed would indeed be necessary to perform the flush operation for each character. Since the flush operation always generates two codewords, this would cause the output stream to be between 125% and 300% larger (for 9 to 16 bit wide codewords) than the input stream. This negative compression is of course not caused by something inherent in the actual LZW algorithm, but merely by the need to immediately flush any data. There are however situations when LZW will generate negative compression, independently of any flushing.’

5. An experimental implementation

The author implemented a data compression layer on top of TCP for the **KA9Q** Internet Package [Karn87] for MS/DOS. The implementation is based upon the original **LZW** specification but with the addition of the *flush*

¹ An example of this is when compressing a file that has already been compressed. To avoid this problem it would be necessary for the application to detect that its data has already been compressed and to disable any further compression by lower layers.

codeword for the flush operation. Another difference is that variable size codewords are used. The application program is able to specify the maximum size of the codewords, which must be in the range of 9 and 16. The encoder begins with 9 bit codewords and **increases** the size as needed. When the limit is reached, a **special dear** codeword is emitted. Both the encoder and decoder will then clear their dictionaries and revert to 9 bit wide codewords. These mechanisms for handling variable size codewords, and for handling the exhaustion of the dictionary are very similar to the mechanisms used by the **LZW** variant in the GIF graphics format [GIF87].

A reason why one would like to limit the maximum size of the codewords is to avoid possible memory limitations at either peer. The **KA9Q** program, for instance, does not leave much spare memory. The experimental implementation provides a tradeoff between memory usage and speed. When minimizing the memory usage, each dictionary entry uses three bytes. This is accomplished by observing that each codeword can be expressed as the string of another codeword followed by a single character. Thus, out of the three bytes, two are used as a pointer to a previous codeword, and the third **represents** the character that should be appended to the string of the previous codeword.

Still, when using only three bytes per entry, 16 bit codewords would require a dictionary size of 192 kbyte. If compression is used in both directions, the memory requirements would be doubled. To prevent accidental memory exhaustion, the encoder and decoder exchange their recommended maximum codeword size values. The smallest of the two values is used by both parties.

The **decoder** can construct its dictionary so that it can quickly look up a string by its codeword. If each entry only consists of one character and a pointer to another codeword, the decoder will get the string in reverse order. But the decoding will **still** be fast. The encoder, on the other hand, must search its dictionary to see if a given string **is** already in the dictionary. This is a very time consuming process when the dictionary grows in size. A way of avoiding this problem would be to restrict the size of the codewords to a **size** that generates reasonably small dictionaries, such as 12 bits. It has also been argued that using codewords larger than 12 bits has **minimal** impact on the efficiency of the

compression. English ASCII text will usually give a compression ratio of 2:1 already at small codeword sizes.

The dictionary can be constructed in different ways to make searching for a string faster. The experimental implementation provides a “fast” mode where a hashing mechanism is used to speed up searching. Dictionary entries do now have to be kept as a linked list. This causes each entry to be 5 bytes, a relatively significant increase in memory usage.

The actual performance of the KA9Q LZW implementation depends mainly on the type of hardware the program is being run on. Using an 8 MHz AT clone there were no problems with keeping a 1200 baud SLIP link saturated when compressing with up to 12 bit wide codewords using either “fast” or “compact” modes of operation.

6. The V.42bis compression standard

The CCITT has published a recommended data compression standard called V.42bis [CCITT90]. It is being implemented as an integral part of many telephone modems. The compression method is in essence LZW, and one is assumed to run it on top of an error free communications link. This is usually achieved on telephone modems by using the V.42 error correction protocol. But V.42bis may also be run on top of some other protocol that provides an error free link, such as TCP.

V.42bis has many similarities with the LZW implementation in the KA9Q package described above. It also uses variable size codewords and a special *flush* codeword for the flush operation.

In V.42bis, the *flush* codeword is always added to the output buffer that is being flushed and then the buffer is padded into an integer number of bytes. This is not entirely necessary, however. The LZW implementation for the KA9Q package uses the *flush* codeword to pad the output buffer into an integer number of bytes. This means that the part of the flush codeword will be in one output buffer and the remainder will be at the beginning of the next output buffer. This works because the *flush* codeword does not need to be processed by the decoder until there are some more codewords to decode. By padding with “blank” bits, V.42bis wastes up to 7 bits (or 3.5 bits on average) per output buffer when compared to the KA9Q

implementation.

CCITT specifications follow the tradition of trying only to specify the operation of the encoder. The decoder should just perform the logical inverse of the encoder. In trying to keep up with this tradition, the CCITT had to modify the way the decoder is assumed to handle the receipt of codewords that it does not yet know about. As described previously, if the input stream consists of the string “cccc”, a standard LZW encoder will encode it as the codewords for “c”, “cc”, and “cc” followed by the flush codeword. The first “cc” will not be in the dictionary of the decoder, but it knows that this can only happen for repeated character strings. The decoder can find out which string the unknown codeword must represent by using a relation between consecutive codewords that holds true in this case. V.42bis, however, differs from the original LZW specification in that the encoder will refrain from emitting a codeword that is identical to the latest codeword it added to its dictionary. (See p. 27 in the V.42bis specification for the exact algorithm.) In this example, the V.42bis encoder will emit the codewords for “c”, “c”, “cc” and “c” followed by the *flush* codeword. This is one codeword more than what would be needed by systems following the original LZW specification (such as the KA9Q implementation.) In many cases, however, both V.42bis and original LZW produce the same amount of codewords for a given string of repeated characters.

There is a provision in V.42bis for the application program to temporarily disable compression. This is very useful when the flush operation would otherwise have to be performed after every single character. (E.g., when transmitting keyboard input in remote echo or full duplex mode.) Without this facility, those cases could lead to a quadrupling of the bit stream, as mentioned previously.

The V.42bis specification includes the data structure, called *trie* [Knuth73], to be used in the dictionary by the encoder. The trie allows for efficient dictionary lookups by using pointers to “child”, “parent” and “sibling” codewords. This causes each dictionary entry to use at least 7 bytes. When the dictionary is full, it is not cleared as in GIF or in the KA9Q implementation², but rather pruned in a circular fashion. The pruning algorithm begins with the oldest codewords, and deletes those that have no children or siblings. The result is very similar to a

Least Recently Used algorithm.

7. Alternative algorithms

LZW derived algorithms are very popular, at least until recently. The LZW algorithm has however recently been patented by Unisys. LZW implementations also risk infringement of the Miller and Wegman patent which is held by IBM. (Miller and Wegman discovered LZW independently from Welch [Miller84].) As far as the author knows, neither of these patents have yet been tried in court

There **are** several other dictionary algorithms based on the original **Ziv-Lempel** methods. Although they may not be as fast or efficient as LZW, implementations of these algorithms do not yet risk patent infringement. There are also several promising statistical compression methods, in particular PPMC [Moffat88]. Although efficient, statistical compression methods often require lots of processing. PPMC seems to be a good compromise and it has been shown to perform well on medium range workstations. [Cate91]

8. Conclusion

Data compression is a viable method for increasing the throughput of a communications link. Care must be taken when choosing a suitable implementation of an algorithm. The implementation may otherwise only decrease the amount of data transferred while imposing delays that lower the throughput. Special consideration must be taken when choosing an algorithm for compressing interactive **traffic**. An LZW implementation for interactive TCP data in the **KA9Q** Internet package has been described. **V.42bis** solves some of the drawbacks of the **KA9Q** implementation, but has also been shown to use less efficient coding than **KA9Q**.

9. References

- Bell89 Bell, T., et al. "Modeling for text compression." *ACM Computing Surveys*, Vol. 21, No. 4, December 1989.
- Cate91 Cate, V., Gross, T. "Combining the Concepts of Compression and Caching for a Two-Level Filesystem." School

of Computer Science, Carnegie Mellon University, 1991.

- CCITT90 "CCITT Recommendation Data Compression **Procedures** for Data Circuit Terminating Equipment (**DCE**) using **Error** Correction Procedures." *Vol. XI, Rec. V.42bis*, Geneva 1990.
- GIF87 "GIF. Graphics **Interchange** Format (tm.)" CompuServe Inc, June 1987.
- Jacobs88 Jacobson, V. "Congestion Avoidance and Control." *Proceedings of Sigcomm '88*. ACM, August 1988.
- Jacobs90 Jacobson, V. "Compressing **TCP/IP** Headers for Low-Speed Serial Links." *RFC-1144*, February 1990.
- Karn87 Karn, P. "The **KA9Q** Internet (**TCP/IP**) Package: A Progress Report" *6th Computer Networking Conference*. ARRL, August 1987.
- Knuth73 Knuth, DE. "The Art of Computer Programming." Vol. 2, "Sorting and Searching." Addison Wesley, Reading, MA, 1973.
- Miller84 Miller, V.S., Wegman, M.N. "Variations on a theme by Ziv and **Lempel**." In "Combinatorial Algorithms on Words." (Apostolico, A., Galil, Z., editors) *NATO ASZ series*, Vol. F12. Springer-Verlag, Berlin 1984.
- Moffat88 Moffat, A. "A Note on the PPM Data Compression Scheme." *Tech. Report 88/7*, Dept. of Computer Science, University of Melbourne, July 1988.
- Storer88 Storer, J.A. "Data Compression: Methods and Theory." Computer Science Press, Rockville, MD, 1988.
- Welch84 Welch, T. A "Technique for **High-Performance** Data Compression." *IEEE Computer*, Vol. 17, No. 6, June 1984.
- Ziv77 Ziv, J., **Lempel**, A. "A Universal Algorithm for Sequential Data Compression." *IEEE Trans. Inf. Theory*, Vol. **IT-23**, No. 3, May 1977.
- Ziv78 Ziv, J., **Lempel**, A. "Compression of Individual Sequences via **Variable-Rate** Coding." *IEEE Trans. Inf. Theory*, Vol. **IT-24**, No. 5, September 1978.

² V.42bis does provide a *clear* codeword, however. But its use is left undefined.

PROPOSED DESIGN AND STRATEGY FOR A RADIO DIRECTION FINDING NETWORK USING DOPPLER ANTENNAS, PACKET, SPREAD SPECTRUM, AND TRANSMITTER SIGNATURES BY DIGITAL SIGNAL PROCESSING

Andrew J. Korsak, Ph.D., VE3FZK/W6
504 Lakemead Way, Redwood City, CA 94062

I. BACKGROUND

Radio Direction Finding (RDF), also commonly referred to simply as DF, has been around for a long time, pretty well ever since radio began. For serious needs such as locating spies during war and malicious or inadvertent interference, some very sophisticated and expensive equipment has been applied. Amateurs have enjoyed the sport of “fox hunts”, or “T-hunts”, as they are frequently called, on a lesser scale. Small, isolated groups of hams have been experimenting with DF equipment and techniques of generally much lower quality than in the professional environment. This generalization applies to most aspects of amateur radio, of course, where cost by far outweighs other limiting factors that professionals can afford to trade off.

II. THE MENACE: JAMMERS, BLOCKERS, CROSS-BANDERS AND KERCHUNKERS

The recent rise in malicious radio interference is noteworthy. This situation calls for immediate, serious endeavours on the part of radio amateurs to police their ranks, at least in highly populated, (yes, also polluted! --RF-wise is not the only way) cosmopolitan areas, such as the San Francisco Bay Area. A number of causes are undoubtedly responsible:

- A. Amateur transceivers have become so advanced that they are often already capable of transmitting outside of their designated ham bands (and nearly always capable of receiving past the band limits).
- B. When not capable of out-of-band transmission initially, they can often be easily modified to do so; in one case I heard of, when the CMOS backup battery is removed, the microcontroller reverts to a boot-up mode wherein it is required to reset the band limits!
- C. No-code licence and the VEC's: these factors have greatly increased the growth rate of our amateur ranks, and it is a highly debatable issue, but the bottom line appears to be that we may be acquiring a greater number of “kooks” among us.
- D. Whereas in the past hams tended to be more technically oriented, the “communicator personality” seems to be the dominant one today, which is not to say that's bad; consider the abundance of valuable ARES contributions. It is my impression that some of the electronics “nerds” among us may be interested

in fooling around with touch-tone access to repeaters that they are not authorized to use, but the "**jammer**" type of personality is more likely to represent some kind of behavioral problem which more easily "slips through the cracks" of the currently easier to obtain amateur **licence**.

III. THE BIG DANGER: WHAT HAPPENED IN SWEDEN

In Sweden, no-code **licencing** began quite a few years ago. Then, after some time, there began to arise a great amount of malicious interference and illegal transmissions. The Swedish equivalent of the US. FCC apparently became so frustrated with tracking down jammers and illegal transmissions that they imposed restrictions on transmitting equipment which, as far as amateurs in Sweden are concerned, boil down to the following:

NO EQUIPMENT THAT CAN BE EASILY ADAPTED TO OUT-OF-BAND TRANSMISSION CAN BE OPERATED OR EVEN OWNED BY AMATEURS!!!

Actually, that may not be such a bad idea. If such equipment is liberally sold, it is bound to be abused. In the **U.S.**, **MARS** operation requires special permits to have equipment modified. In Sweden, apparently all sellers will lose their permits to distribute equipment if they are caught supplying such equipment as described above.

In fact, the "Swedish **FCC**" apparently pulled off a real "sting" operation. They appeared at a ham gathering on one occasion and offered to test the quality of the hams' equipment. They convinced the hams to leave all their equipment on a table, and then they proceeded to test each unit for potential out-of-band transmission. Any units that "passed" the test were **CONFISCATED!!!**

Needless to say, they got away with that trick only once. The amateur grapevine travels just as fast over there as anywhere! And that's not the worst of it; apparently, what the above law means is that

NO HOMEBREW TRANSMITTING EQUIPMENT IS ALLOWED IN SWEDEN

because of the preemptive "bottom line" restriction above. I wonder how the "legal beagles" in our amateur communities of North America would deal with that kind of imposition!

The question is: **DO WE WANT THIS TO HAPPEN IN THE UNITED STATES, OR CANADA (or anywhere else)?**

If you agree that something serious needs to be done to eliminate abuse of our amateur bands, I would appreciate hearing from you on any ideas you may have, and I hope this paper gets some groups taking some action elsewhere.

IV. THE PROPOSED PLAN

What I am embarking on has two main aspects, individual **DF'ing** and networked

cooperation. Emphasis will be on an improved design for the Roanoke type of Doppler antenna implementation to be used at either individual fixed/mobile DF units or the proposed network “system nodes”.

A. Improved Hardware/Firmware and Methodology at DF Units

1. Improving individual fixed/mobile DF capability

The traditional DF’ing loops, portable beams, switched dual rubber ducks, body shielding an HT with no antenna when close to a “fox”, etc., are all fine as an add-on capability, but chances are the bad guys are going to be too cagey, as they have been recently in the San Francisco Bay Area, for such methods to be adequate. What is needed is an instant response capability.

I am revising the so-called Roanoke Doppler antenna system as described in the radio direction finding handbook published by TAB Books [1]. That design is somewhat out of date and a microcontroller such as the Intel 8751 can handle the job more flexibly. In addition, once you have the power of this microcontroller, why not utilize its built-in UART to pass data and control to and from other DF system components?

The Roanoke Doppler antenna system operates on the principle of electronically simulating physical rotation of a vertically polarized ground plane antenna in a horizontal plane, by switching on only one of four antennas at a time. This is accomplished using diodes, RF chokes, feed-through capacitors and quarter-wave coax sections to isolate the diodes from the antennas and the feed point. Control and data acquisition is achieved by a combined function circuit that cycles through switching of the diodes! at 8 KHz while simultaneously filtering out the resulting superimposed 2 KHz audio tone, using a concurrently switched capacitor filter and some op amps with feedback for amplitude regulation. The tone then passes into a positive-edge zero-crossing detector followed by determination of the time of the waveform peak relative to the base 2 KHz clock cycle derived from the 8 KHz that is switching antennas and capacitors in the filter. Relative to the 2 KHz clock, direction of the RF wavefront is then indicated by turning on one of typically 8 or 16 diodes in a circular display arrangement, which may be calibrated by rotating the antenna platform on top of a car, for example.

There are a number of improvements that I am investigating:

- a. Using the 8751 microcontroller to generate the antenna switching pulses and switching the capacitors by utilizing some of its 32 I/O port lines
- b. Generating a “spread spectrum” like effect in place of the 2 KHz audio tone, i.e., using something like a Gold or Barker code, spreading the switching effect all over the audio spectrum, so that there would not be just a single audio tone that may get wiped out by modulation on the received signal.
- c. Providing for some multipath recognition capability by taking into consideration

more than just a phase angle of a supposed pure sine wave resulting from a single RF wavefront being phase modulated by antenna “rotation”, i.e., some further attributes of the resulting modulation need to be recorded, e.g. binary code switching sequences being “distorted” into other patterns whose parameters quantify something like an antenna pattern profile; this obviously ties in with the “audio spread spectrum” idea.

2. Providina network nodes with advanced DF features

At a number of San Francisco Bay Area hill top sites, with the cooperation of repeater groups that will be undoubtedly keen to help discourage malicious jamming, etc., there will be Doppler antennas connected to secondary remotely controllable receivers. These receivers will comprise a special set of “system nodes”, and will be vectored simultaneously onto a frequency to be monitored upon request, tentatively by packet in the initial design.

At the system nodes, there will be installed copies of the **8751** based controller I am designing, which will tune the monitoring receivers at these sites and send out, over a packet network, the observed bearing, along with a time tag and any other available information.

Additional features being contemplated:

a. Sianature analvsis

A “signature” of the transmitter being monitored can be determined. Apparently, all transceivers coming off a given production line share common characteristics with regard to how their **PLL’s** lock on over time from initial **turn-on**, e.g. creating a particular “signature” in terms of spectrum sweeping as they lock on.

b. Uploadina data from mobile/portable laptops

Information will be recorded at one or more sites where DF teams can access it to assist in localizing a malicious jammer, for example. As team members close in, they can upload further more accurate bearings using mobile packet with lap top **PC’s**.

c. Spread spectrum for the control links

If the “bad guys” get nasty and start interfering with the operation of this system, we may need to resort to spread spectrum on 440 MHz or higher for the control links.

d. Diaital maps from USGS to help locate the “bad guys”

In fact, some of our DF teams may even use one of those new car navigation systems, which will direct them on an optimal path toward a jammer.

e. Digitized audio announcement of map coordinates

Wouldn’t that be slick! Perhaps a jammer will become so embarrassed that he will get off the frequency! If he continues to play games, the DF teams will

simply find it easier to locate him.

f. Civil lawsuits for deprivation of repeater use

If the FCC or the Federal Government cannot not slap an adequate penalty on a jammer, some of us that are affected should technically “fine” the jammer ourselves, using the civil courts, on the grounds of interfering with our collectively arranged phone patch privileges, for example.

V. ACKNOWLEDGEMENTS

Thanks are due to a number of amateurs with whom I have discussed this project so far, who supplied some of the ideas: Lars Karlssen, **AA6IW**; Randy Roberts, **KC6YJY**; Glen Tenney, **AA6ER**; Rick Gilbert, **WB6ADB**; Fred Pearlman, **WDØDLM**.

VI. REFERENCES

1. Transmitter Hunting, Radio Direction Finding Simplified., Joseph Moell, **KØOV** and Thomas N. Curlee, **WB6UZZ**, TAB Books, 1987, Chapter 9, pp. 120-141

DESIGN AND IMPLEMENTATION OF CELP SPEECH PROCESSING SYSTEM USING TMS320C30

A. Langi, VE4ARM and W. Kinsner, VE4WK

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba, Canada R3T-2N2
E-mail: Kinsner@CCM.UManitoba.CA
Radio: VE4WK@VE4KV.MB.CAN.NA

Abstract

This paper presents a design and implementation of a signal processing system for telephone-quality speech transmission through a low bit-rate channel, such as **packet-radio**, at rates as low as 4.8 **kbit/s** in either real-time or off-line mode. The system compresses speech signal down to 4.8 **kbit/s** using the code-excited linear predictive (CELP) coding scheme adapted from the U.S. Federal Standard **FS-1016**. The system implements the CELP scheme using a floating-point digital signal processor (**DSP**) to perform real-time, interactive (full- or half-duplex) or fast off-line network-based applications. The system is implemented on a low-cost personal computer (PC) equipped with a **TMS320C30** evaluation module (**EVM**).

1. INTRODUCTION

Current voice systems are no longer sufficient to meet demands on high quality voice communications over long distances. Present systems, operating on HF, VHF, and UHF, use analog signals for speech transmission [Youn90], [Blo90]. However, the quality of analog voice deteriorates rapidly over long distances. In VHF/UHF transmission, the voice usually becomes too noisy after only a few repeaters. Transmission of digital signals is considered superior to its analog counterpart due to the ability to completely reconstruct the signal at each repeater. This ability results in a higher immunity to noise [Kins89], [Blo90]. In addition, the digital transmission can use error protection schemes that increase the robustness of the transmitted signals.

The main problem of digital voice transmission through radio channel is the bandwidth limitation of the channel. Direct digitizing of the speech for telephone quality results in digital data with a rate of 64 **kbit/s**. This rate is too high for the channel, and, consequently, the system needs a speech compression scheme to bring down the bit rate.

A speech compression scheme called code-excited linear predictive (CELP) coding is very attractive for digital speech

communication using radio channel [ScAt85]. The coding can compress speech down to a rate of 4.8 **kbit/s**. The resulting speech quality is better than any of speech compression schemes at the same bit rate. The U.S. Government has adopted the scheme as the proposed Federal Standard 1016 (FS-1016) for digital radio as well as secure voice communication at a rate of 4.8 **kbit/s** [CaTW90].

Another problem of digital radio communication is how to protect the compressed speech from the channel noise. Forward error correction (**FEC**) schemes can be used to protect the compressed speech. The FS-1016 CELP uses FEC schemes including a Hamming code. More sophisticated codes can be used such as in the proposed **FS-1024** for land mobile radio system. However, the utilization of the codes results in higher bit rates, such as 8 **kbit/s** for the proposed FS-1024 system [RTWC89].

We have designed and implemented a CELP speech processing system that can be used for voice communication through packet radio. This system is capable of real-time, full-duplex speech communication at a rate of 4.8 **kbit/s**. It also can be used to support non real-time applications such as voice mail (store-and-forward mode). The system includes a forward error protection scheme to increase the robustness of the speech during the transmission. The system is implemented on a low-cost personal computer (PC-AT) equipped with a **TMS320C30** evaluation module (**EVM**). The system maintains interoperability with other FS-1016 systems. It is also an extension of another implementation on the NEC 77320 EM [Laki90], [Laki91].

2. SYSTEM ARCHITECTURE

2.1 Basic Scheme

Figure 1 shows the system architecture corresponding to a CELP system [LaKi90]. The system consists of a *controller*, a *data processor*, and *data media*. The controller

manages the timing of the process and the flow of data. The data processor consists of functional blocks, transforming speech data into several required forms for performing speech compression, speech decompression, and error protection processes. The functional blocks in the upper section constitute the CELP **transmitter**, while the blocks in the lower section constitute the CELP **receiver**. Finally, the data media are storage device and packet radio channel. The speech data can be stored or transmitted on the data media.

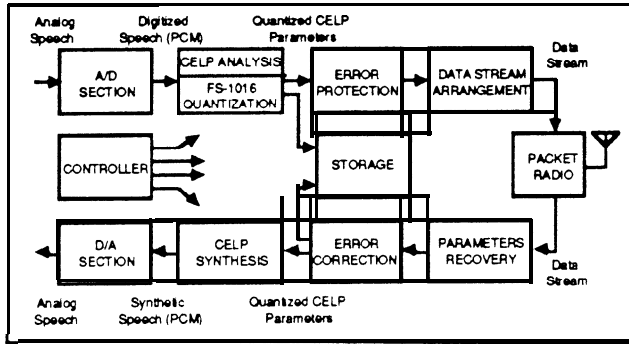


Fig. 1 System architecture for implementation.

The data processing of CELP transmitter begins with converting analog speech signal coming from a microphone and an amplifier, into a pulse code modulation (PCM) form by an analog-to-digital (A/D) converter. The A/D converter operates at an 8-kHz sampling rate and 14-bit resolution. The block called CELP analyzer compresses the digitized speech into speech data called CELP **parameters**. An FS-1016 quantizer enables low bit-rate representation of the CELP parameters and standardizes the bit definition of the parameters. A forward error protection scheme adds redundant bits for protecting the quantized CELP parameters from channel noise. The protected CELP parameter bits are reordered according to the FS-1016 protocol, and then applied to a packet radio. A user has an option to store the quantized CELP parameters on a storage device.

On the other hand, the CELP receiver reverses the data processing. The CELP parameters are recovered from a data stream coming from the packet radio. An error correction scheme detects and corrects data errors that are within its correction range. A CELP synthesizer uses the corrected CELP parameters to produce speech in a PCM form. A digital-to-analog (D/A) converter converts the PCM data into an analog form that can be used for speech listening using an amplifier and a speaker.

The main functional blocks of the data processing are the CELP analyzer and synthesizer. The two blocks, originally developed by **Atal** and Schroeder [ScAt85], enable the compression of **PCM** speech data into **CELP** parameters and the decompression of the CELP parameters into PCM speech data. Since the CELP analyzer utilizes a CELP synthesizer due to an analysis-by-synthesis approach, we **first** describe the CELP synthesizer. In addition, it is easier to understand the role of each CELP parameter using the description of CELP synthesizer.

2.2 CELP Synthesizer

Figure 2 shows a scheme in the CELP synthesizer. The synthesizer uses CELP parameters to produce PCM speech data. As shown in Fig. 2, the CELP parameters are linear predictor (LP) filter coefficients, adaptive code book (ACB) entry and gain factor, and stochastic code book (SCB) entry and gain factor. The parameters control three blocks, e.g., the LP filter, the ACB, and the SCB, to produce speech.

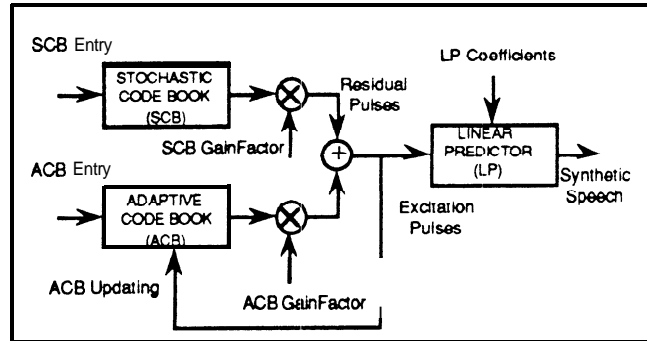


Fig. 2. CELP synthesizer.

The scheme is closely related to a human speech production system. The LP filter represents a human vocal tract. A change in the shape of the vocal tract during speech generation results in a change of **frequency** characteristics of the vocal tract. The LP filter (an all-pole adaptive digital filter) performs this frequency characteristic change by changing its coefficients. The speech is produced by applying signals called excitation pulses to the LP filter.

Speech has a regular fundamental frequency, called **pitch frequency**, which introduces **tone to the** speech. The characteristic of the speech with regular pitch frequency is the speech waveform **periodicity**. The ACB, that is a code book containing many sequences of previous excitation pulses, can preserve the periodicity by supplying a proper

sequence of previous excitation pulses. The proper sequence is selected by an ACB entry and the amplitudes of the sequence elements are scaled by an ACB gain factor. It should be noted that the pulses stored in ACB are periodically updated with the latest excitation pulses.

The SCB is a code book storing many sequences of pulses, called residual pulses. The pulses are called residual because they can be seen as the remainder of speech samples after vocal tract and pitch frequency informations are taken away. In contrast with the ACB pulses, the residual pulses are fixed, normalized, and obtained experimentally. We use the SCB defined by the proposed FS-1016. An SCB entry selects a particular sequence, and an SCB gain factor scales the amplitude of the normalized pulses. The resulting residual pulses become the excitation pulses after being combined with the ACB pulses.

In summary, the CELP parameters control the CELP synthesizer to produce a particular speech waveform. Different parameters result in different characteristics of speech. Thus, a set of the CELP parameters can be seen as a speech representation. A speech compression is achieved because the CELP parameters require much fewer representation bits. Using the proposed FS-1016 CELP, we can represent 240 speech samples (PCM) in 144 bits of CELP parameters, as shown in Table 1.

Table 1. Bit allocation of proposed FS-1016 CELP system.

TYPE	ALLOCATION	TOTAL
Linear Predictor	3,4,4,4,4,3,3,3,3	34
Adaptive CB	Entry: 8+6+8+6 Gain: 5 x 4	48
Stochastic CB	Entry: 9+9+9+9 Gain: 5 x 4	56
Synchronization	1	1
Future Expansion	1	1
Error Correction	4	4

2.3 CELP Analyzer

Figure 3 shows a CELP analyzer. The analyzer compresses the incoming speech into CELP parameters. A block called LPC analysis obtains LP parameters from the input speech. The LPC analysis utilizes a scheme of coefficient estimation/tracking of an all-pole, time-varying filter [Pars86].

The SCB and ACB parameters are obtained using an analysis-by-synthesis scheme. An error minimization block

applies every possible combination of the SCB and ACB parameters to a CELP synthesizer. For every set of parameters, there is a corresponding error signal. The set giving the minimal magnitude of error is chosen as the output of the analyzer. A perceptual **weighting (PW)** filter weights the error signal **prior** to the magnitude measurement to compensate nonuniform perceptual effects of error on **different speech frequencies**.

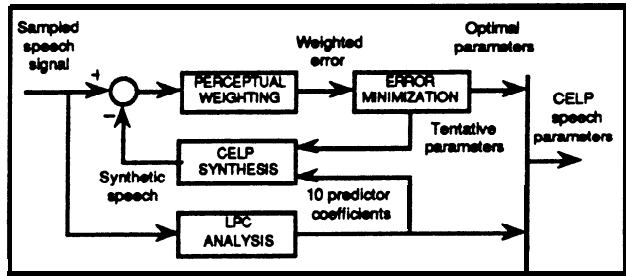


Fig. 3. CELP analyzer.

In practice, an analyzer's scheme may be different from the one described due to the implementation of fast algorithms, such as a **joint optimization** scheme used in our system [CaTW90]. However, every scheme is derived from the original scheme shown in Fig. 3.

3. SYSTEM IMPLEMENTATION

The system is implemented on a PC-AT equipped with a TMS320C30 EVM. The EVM contains a digital signal processor (DSP) and an analog interface. This platform allows us to implement **all** the blocks of Fig. 1 (except for the packet radio). The CELP algorithms are then coded to run on the EVM and the host PC-AT for performing functions described above.

3.1 Hardware Implementation

All functional blocks are organized for implementation according to a given hardware platform.

Hardware Platform

The host computer has an INTEL 80286 processor, 512 Kbytes memory (minimum), a hard disk, and a serial port (RS232C). The EVM is a half-size board that is installed in a PC-AT slot. The EVM contains a powerful floating point, **32-bit DSP** from Texas Instrument (TMS320C30) that has 60 ns instruction cycle. The DSP has 2K words of internal RAM, while the EVM has 16K words of zero-wait

state static RAM for codes and data.

The EVM is equipped with an analog interface circuit (AIC) based on the TLC32044. The TLC32044 is a programmable, 14 bits A/D and D/A in a single chip. It also has a programmable filter and sampling rate controller.

To communicate with its host, the EVM has 16-bit bidirectional ports. At the EVM side, the data interchanges can be driven by either interrupts, or status polling, or direct memory access (DMA). At the host side, the interchanges are driven by status polling.

Block Distribution

The hardware platform influences how the functional blocks are implemented. The A/D and D/A sections are implemented using the AIC of the EVM. The CELP analyzer and synthesizer are implemented on the EVM because they, especially the analyzer, require very high computational power. They are also physically close to the AIC. The error protection, data stream arrangement, and the data media interfaces are done on the host computer. The storage device is the host hard disk. The packet radio must be connected to the host serial port. Finally, the controller is implemented on the PC-AT. The PC-AT then has a user interface on which we can type in commands to choose functions the system has to perform.

3.2 Software Organization

Since some functional blocks have been implemented in hardware, the remaining blocks must be developed in software. However, since the error protection, correction, and data arrangement are look-up table schemes, as well as the media interfaces are standard disk operating system (DOS) procedures, we concentrate on the controller, CELP synthesizer, and CELP analyzer.

Controller

Figure 4 shows a flowchart of the controller. The program begins with initializations of the system including the serial port and the EVM. It also sets the system state to a default setting. The program displays a menu, then wait for a command from a user, and execute the subprograms related to the command.

The most critical and challenging operation mode is the real-time, full-duplex communication mode. We concentrate on this mode, because the other modes are simple modifications of this mode. Figure 5 shows an

approximate schedule of launching PC-AT and EVM subprograms during the real-time, full-duplex mode. Notice that the process is always repeated every frame of 240 speech samples,

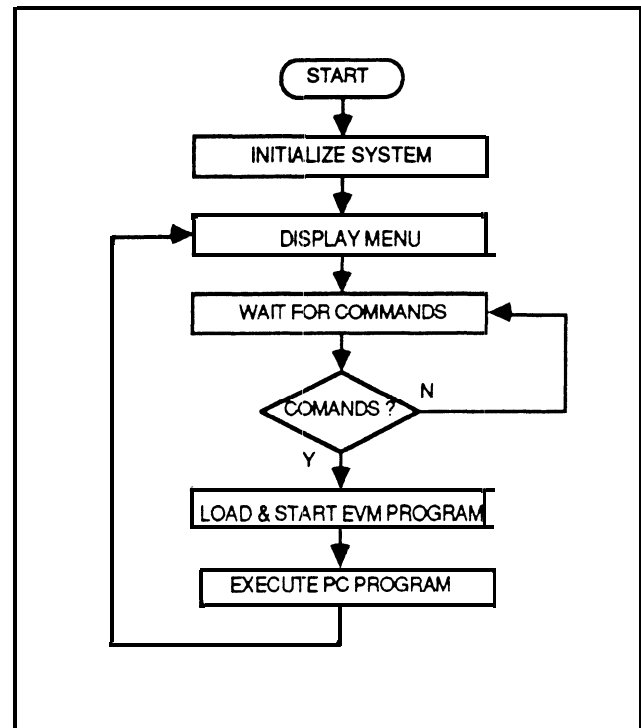


Fig. 4. A flowchart of the controller.

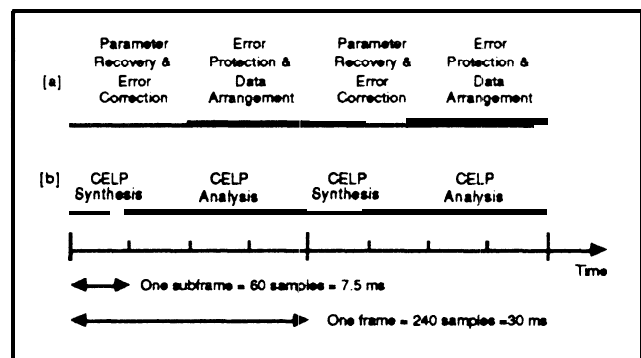


Fig. 5. An approximate schedule of launching subprograms in real-time, full-duplex mode for (a) PC-AT, and (b) EVM. There are interrupts during the execution of the subprograms for analog and serial RS232C I/O on the EVM and PC-AT, respectively. There are also data exchanges between the PC and the EVM at the end of a frame.

The program runs in both foreground (interrupt driven) and background. The main program runs in the background. It consists of a CELP synthesizer and a CELP analyzer.

They are responsible for converting speech data within the given period. The source and destination of the data are several **working buffers**.

On the other hand, interrupt services run in the foreground. An interrupt occurs in the EVM every 125 μ s according to the sample frequency of the A/D. The interrupt service gets a sample from A/D, stores it on a working buffer, takes another synthetic speech sample from another working buffer, and supplies the data to the D/A. Another interrupt also occurs in the PC-AT for serial **RS232C** communication. Here, its service also does similar response, except the data are the CELP parameters and the port is the **RS232C**.

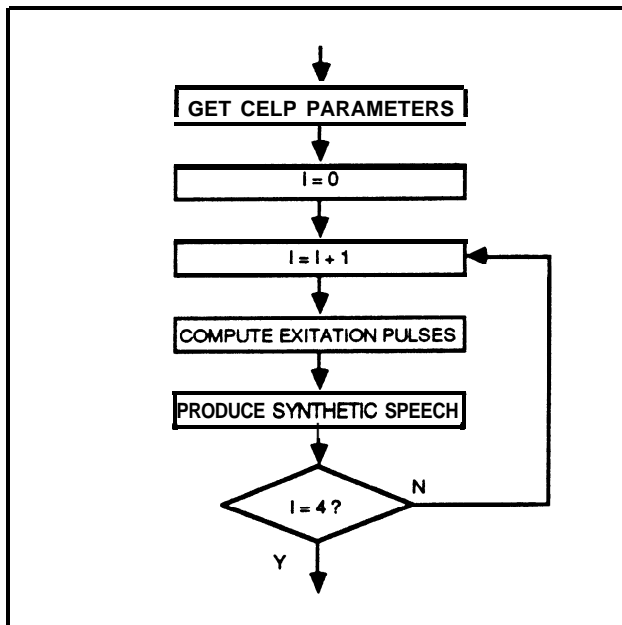


Fig. 6. A flowchart of a CELP synthesizer.

CELP Synthesizer

Figure 6 shows a flowchart of a CELP synthesizer corresponding to the scheme in Fig. 2. The program reads the code book parameters from the working buffer and use them to produce 60 excitation pulses. The content of the ACB is updated using the new excitation pulses. A set of LP coefficient is obtained from the previous and current LSP according to an interpolation rule, and is used to replace the LP filter coefficients. The new LP filter then receives the excitation pulses, and produces 60 samples of synthetic speech. The speech is stored in another working buffer. The process is repeated three more times until 240 synthetic samples have been obtained in the working buffer.

The program then returns to the controller, and the controller launches the CELP analyzer.

CELP Analyzer

Figure 7 shows a flowchart of a CELP analyzer corresponding to the scheme shown in Fig. 4 with a slight modification. Here, the PW filter is cascaded with the CELP synthesizer block. The LP filter inside the CELP synthesizer can then be combined with the PW filter. A duplicate of the PW filter is placed before the summation block to compensate the change.

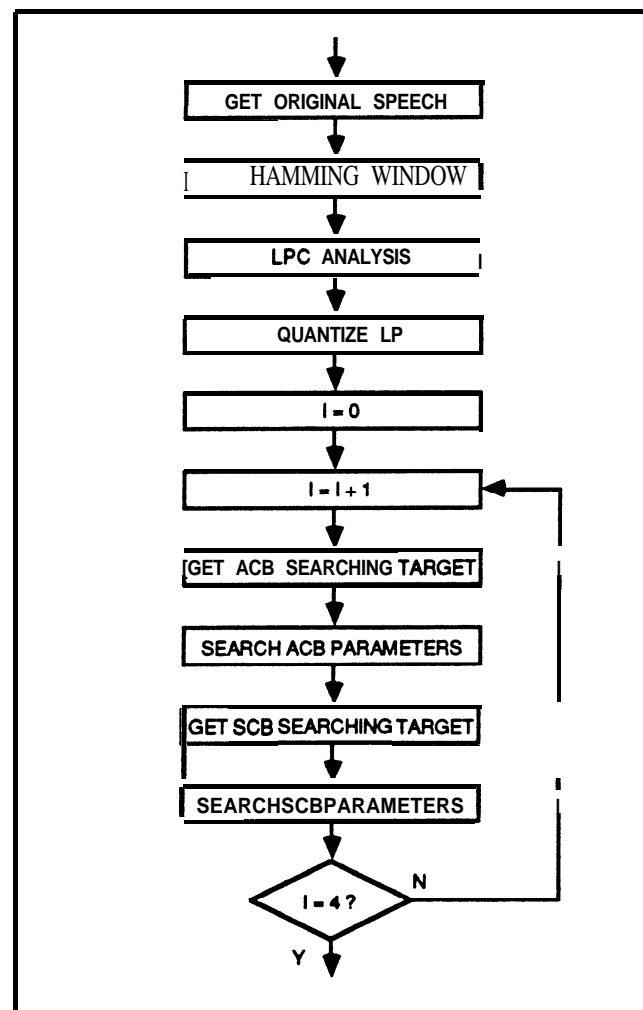


Fig. 7. A flowchart of a CELP analyzer.

The program reads the speech data from the working buffer and weights the data using Hamming window [Pars86]. The program then performs LPC analysis to obtain the LP coefficients. The LP coefficients are quantized into a line spectrum pair (LSP) form [SoJu84].

The quantized LP coefficients are used in the LP filter for finding code book parameters.

To obtain ACB parameters, we must first provide the ACB searching target. The program computes the zero response of the new LP filter and subtracts the zero response from the original signal to compensate ringing from previous frames. The resulting signal is applied to the PW filter to obtain the ACB searching target. A code book searching subroutine (joint optimization) is then called to find the ACB parameters.

To obtain SCB parameters, we must also provide the SCB searching target. The ACB parameters obtained above is used to create excitation pulses. The program applies the pulses to the cascade of LP and PW filter. The resulting signal is subtracted from the ACB searching target mentioned earlier to have the SCB searching target. Having the searching target, the program calls joint optimization subroutine to find the SCB parameters.

The joint optimization scheme automatically quantizes the gain factors. The code book entries are not quantized except for the ACB entries. For each even subframe, the ACB entry is represented by the offset of the actual entry relative to the previous ACB entry.

4. SYSTEM DEVELOPMENT

The use of the EVM makes the system development much easier because the EVM is accompanied with powerful development tools. However, applying careful development methodology is still critical due to several problems. First, fast CELP algorithms are not trivial, and involve many computational processes. Second, the EVM has a limited size of installed memory, resulting in a need to compact the program size and to schedule the use of working buffers. Third, the real-time requirement demands a way of coding every function to achieve execution time as short as possible. Finally, since the TMS320C30 itself is a processor with pipelining and parallel execution, a higher complexity of software must be considered. System development begins with hardware setup, software development, and finally system test.

4.1 Hardware Setup

The final system is a PC-AT equipped with the EVM, a two channel amplifier, a microphone, a speaker, a TNC, and a transceiver. For full-duplex implementation the TNC

and the transceiver must be set accordingly. However, a smaller configuration can be used during the development process. A PC-AT is sufficient to code and test EVM routines because the EVM development system has a TMS320C30 simulator. The results can be used for real-time debugging and testing on a PC-AT with the EVM installed.

4.2 Software Development

The algorithms are first simulated on a SUN workstation using C programming. This simulation is essential because the algorithms are complex. The C program is gradually converted into modular subroutines according to the architecture and flowcharts mentioned earlier. Each subroutine module is optimized and, one by one, recoded into TMS320C30. Each module is then tested before being tailored into the EVM program.

There are two ways to convert a C module into its TMS320C30 counterpart. First, the EVM development tools have a C compiler that automatically transfers a C program into a TMS320C30 program. Second, we can write a TMS320C30 program directly with the help of the logical structure and data type a program has. Although we can have full control and very efficient codes using the latter approach, the former approach is more convenient. We utilize both by using the first method to have a draft of code, and using the second method to compact the draft. However, for simple control and time critical process, such as interrupt handling, we use the second method.

Each routine is tested off line using a TMS320C30 simulator. Here, we can check the number of clocks and registers used. In addition, we can control several options, such as pipelining, for easier debugging. A real-time debugger is then used to test the program in actual processor. Here, we can fine tune the software and test the peripheral specific functions that are not fully supported by the simulator. Finally, the verified codes are loaded into the EVM for normal operation or operational tests.

4.3 Preliminary Test

Simple tests have been performed to verify the system. We connected the serial port of the PC-AT to another computer at a rate of 4.8 kbit/s. The other computer received the CELP parameters and immediately returned the parameters back to the PC-AT. The PC-AT reconstructed the speech from the parameters for a listening process. This

test did not cover the effect of the channel noise to the system performance, nor the practical problems that may be encountered in a real application using a TNC and a transceiver. However, it did verify the speech compression capability and quality of the system.

A study to enable the system to be used for mobile communication is desirable for future work. In the future, this capability may be increasingly of interest. The study involves the finding of good error protection schemes, especially to face noise of multipath fading channels.

5. CONCLUSIONS

The design and implementation of a speech processing system for communication through packet radio have been described. The system implements CELP coding scheme to achieve high speech quality in a bit rate as low as 4.8 kHz. It is implemented on a low-cost PC-AT equipped with a TMS320C30 EVM. It is capable of real-time, full-duplex operation, and can be modified for other modes. Although the CELP algorithms are complex, the system is developed easier using the powerful tools accompanied with the EVM and the development method for coding a signal processing application. A simple test verifies the speech compression and quality of the system. Future work and tests are still needed to enhance the system for practical and advanced applications, such as voice mail.

ACKNOWLEDGEMENT

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Inter University Centre (IUC) on Microelectronics, ITB, Indonesia, and Signals and Systems Laboratory, Department of Electrical Engineering, ITB, Indonesia. Special thanks go to Dr. R. McLeod for providing the TMS320C30 development facilities.

REFERENCES

- [Bloo90] J. Bloom, "Considering next-generation amateur voice systems," *ARRL/CRRL 9th Computer Networking Conf.*, pp. 10-15, 1990.
- [CaTW90] J. P. Campbell, Jr., T. E. Tremain, and V. C. Welch, "The proposed Federal Standard 1016 4800 bps voice coder: CELP," *Speech Technology*, pp. 58-64, Apr./May 1990.
- [Kins89] W. Kinsner, "Speech recording and synthesis using digital signal processing," *CRRL Convention*, Winnipeg, MB, Canada, 21 pp., August 1989.
- [LaKi90] A. Langi and W. Kinsner, "CELP high-quality speech processing for packet radio transmission and networking," *ARRLICRRL 9th Computer Networking Conf.*, pp. 164-169, 1990.
- [LaKi91] A. Langi and W. Kinsner, "Code-excited linear predictive speech processing for digital transmission and storage," *Proc. the IEEE Western Canada Conference on Computer, Power, and Communication Systems in a Rural Environment*, IEEE 91CH2927-2, pp. 205-209, 1991.
- [RTWC89] D. J. Rahikka, T. E. Tremain, V. C. Welch, and J. P. Campbell, Jr., "CELP coding for land mobile radio applications," *Proc. Int. Conf. Acoustics, Speech & Signal Processing*, IEEE CH2673-2/89, pp. 465-468, 1989.
- [ScAt85] M. R. Schroeder and B. S. Atal, "Code-Excited Linear Prediction (CELP): High quality speech at very low bit rates,*" *Proc. Int. Conf. Acoustics, Speech & Signal Processing*, IEEE CH2118-8/85, pp. 937-940, 1985.
- [Youn90] N. R. Young, "A discussion of the issues and technologies for HF voice communication," *Proc. Canadian Conference on Electrical and Computer Eng.*, EIC, pp. 68.2.1-68.2.4., 3-6 Sept. 1990.

Digital Networking With the WA4DSY Modem - Adjacent Channel and Co-Channel Frequency Reuse Considerations

Ian McEachern VE3PFH
515 Rougemount Cres.
Orleans, ON Canada K4A 3A1

ABSTRACT

The explosion of digital communications in the world and the continued growth of amateur packet radio, networking with high speed modems is becoming a reality. The Dale Heatherington WA4DSY modem [1] has given the Amateur Radio world the means of implementing the high speed networks with a 56 kb/s Minimum Shift Keying (MSK) modem. However, little is known about the performance of the modem when it is closely spaced, in frequency and proximity, with wide band and narrow band carriers. This paper investigates the performance of the WA4DSY modem with adjacent and co-channel interference for the purpose of providing tools to network planners, frequency co-ordinators and average users, for use in planning and implementing high speed digital networks.

1. Introduction/Background

1.1. The Digital World

The world of communications is going digital. With high speed ISDN systems replacing analog telephone networks, digital television compression replacing analog FM distribution of television signals on satellites, digital High-Definition-Television and digital cellular telephone systems in North America and Europe soon to be introduced, the real world is digital now. Amateur Radio has started going digital as well, but with the technology becoming available from the new developments, it is inevitable that amateur radio will begin experimenting with more digital technology. Some of the applications which will be vying for spectrum in the new digital amateur service will be:

- digital voice repeaters using digital cellular technology;
- digital FSTV at 1.5 Mbps using digital compression techniques;
- digital SSTV at 56 kb/s (or less) using digital video conference and digital compression techniques;
- high speed packet networks; and,
- digital voice and data trunking for inter-city links

With all of the possible applications for digital communications in the amateur service, the very limited allocations for wide band data transmissions and digital voice repeater systems in the future, the task of coordinating the networks to

avoids mutual interference appears daunting.

1.2. Ottawa Digital Network Expansion Plans

In the Ottawa area a 56 kb/s Metropolitan Area Network (MAN) has been in operation for some time[2]. New users have continued to join the network and two of the area PBBS's are on the MAN. A decision was made that a second digital repeater, using the Doug Heatherington, WA4DSY 56 kb/s modem (hereafter called the DSY modem), would soon be required. The existing repeater is a cross-band full-duplex bit-regenerating repeater that receives on 220 MHz (Canadians still have access to the 220 to 222 MHz band) and transmits on 430 MHz.

This decision, however, included two key questions:

- how close could the two high speed carriers be placed on both the transmit frequency and the receive frequency; and,
- how close could they be to local Narrow Band FM (NBFM) voice and packet carriers.

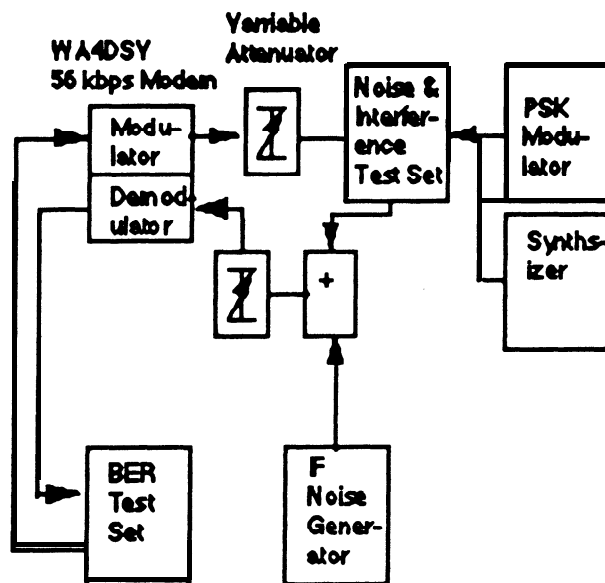
This paper investigates the effects of co-channel and adjacent channel interference due to NBFM and wide band digital carriers for the purpose of determining the minimum channel frequency spacing and network distance spacing, and provides tools for network planners to use in coordinating digital networks.

2. Test Set Up

The test set up used is shown in Figure 1. The author was fortunate to have access to the set-up, which is normally used for testing and characterizing satellite data modems in the presence of interference, at the company where he is employed.

Figure 1

Test Set-Up



Bit-Error-Rate (BER) was the measure of performance of the modem. The BER was measured using a BER Test Set which sends a sequence of random bits to the modulator and then measures the number of incorrect bits received from the demodulator. The output signal from the modulator was combined with noise generated by the Intermediate Frequency (IF) Noise Generator and was fed back to the demodulator. It was found that the demodulator was fairly sensitive to the power level received so a variable attenuator was placed at the input to the demodulator so that the signal could be attenuated when necessary. By varying the output level of the modulator by using an external Variable Attenuator the Energy-Per-Bit-to-Noise-Density-Ratio (E_b/N_0) could be varied.

For the NBFM tests a synthesizer was used to generated an FM modulated signal with 5 kHz deviation and a 1 kHz tone as the interfering

signal. The signal was inserted in the Noise and Interference Test Set where the Carrier-to-Interference-Ratio (C/I) could be set and maintained. The signal from the Noise and Interference Test Set was added to the thermal noise and then to the demodulator.

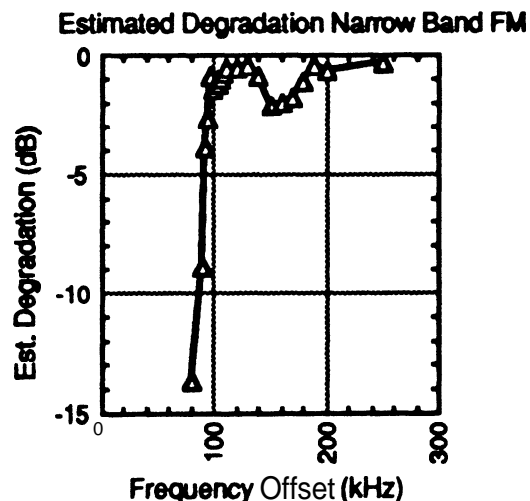
For the wide band digital interference a commercial Phase-Shift-Keyed (PSK) satellite modem was used as the interfering carrier. A second DSY modem was not available at the time the tests were performed so the PSK modem was substituted. However the PSK modem was of similar bandwidth and similar spectrum shape compared to the DSY modem, for the main lobe.

3. Nominal Performance

The tests were performed to determine the degradation due to the presence of the interference. This requires that the nominal performance be characterized. In an ideal world the only degradation on signals is thermal noise which is introduced into the receiving system. It is possible to simulate both the ideal world and the real world by performing tests on an 29 MHz IF loop-back basis. The affects other Radio Frequency (RF) equipment will not be significant on a properly aligned system.

The performance measured was on a modem the author was building up for use on the Ottawa MAN. The actual performance measured may not conform to what other users achieve. The tune up procedures described in the modem Manual were followed, but no attempts to optimize performance

Figure 2



were made. However since the important point of the experiment is to determine typical relative performances, the absolute measurements are not as significant, since all conclusions will be based on relative performance.

Figure 2 shows the BER versus E_b/N_0 performance of the DSY modem used in these tests and is considered the baseline performance.

For the tests that follow the BER is measured and is compared to Figure 2 where an "equivalent E_b/N_0 ", $(E_b/N_0)_{eq}$, is determined. The $(E_b/N_0)_{eq}$ is called equivalent because the interference is not noise, but causes a BER equivalent to an E_b/N_0 . The $(E_b/N_0)_{eq}$ may be used to determine the degradation in performance in dB and, as well, and equivalent interference noise. The equivalent interference noise could then be used to add to link budget calculations for "Veal" systems, and thus enable the network planner to determine the approximate performance expected based on the level of interference expected.

The BER's measured above includes the on-board modem scrambling. The descrambler, which is a shift register and nor gates, tends to multiply errors. That is if the demodulator makes one bit error the descrambler will output a multiple number of errors. However scrambling is necessary to spread the energy evenly over the bandwidth of the carrier and is recommended for use, so it was used for all tests performed.

4. Performance With Interfering NBFM Carriers

In many band plans wide band digital modes are adjacent to NBFM voice and NBFM packet. These tests will give frequency coordinators and wide band data users tools to use in determining frequency plans for wide band digital systems. It should be noted that no attempt has been made to characterize the interference from the wide band data into NBFM in these tests.

4.1. Adjacent Channel Interference

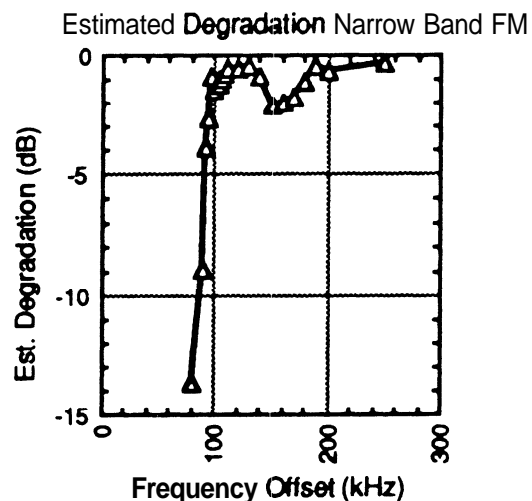
The C/I was set at -10 dB (the interfering carrier had 10 dB more power than the wanted 56 kb/s carrier at the demodulator input) and a nominal E_b/N_0 of 18.5 dB-Hz and the interfering carrier was

stepped from 250 kHz offset. The BER was measured at each frequency and is given in Table 1. From Figure 2 an $(E_b/N_0)_{eq}$ was determined and a degradation from nominal was calculated. The Block Error Rate (BLER) was also measured for 1000 bit blocks, approximately the same number of bits as a typical packet used in Amateur Radio. The BLER would be a reasonable estimate of rate of lost or discarded packets.

Table 1
NBFM Adjacent Channel Interference

Freq. (kHz)	BER	BLER	$(E_b/N_0)_{eq}$ (dB-Hz)	Degradation (dB)
250	1.10E-05	0.30%	18.2	-0.3
200	2.60E-05	0.49%	17.8	4.7
190	1.50E-05	0.36%	113.1	-0.4
180	7.15E-05	1.50%	17.3	-12
170	2.38E-04	4.67%	16.7	-1.8
160	3.60E-04	6.90%	16.5	-2.0
150	4.40E-04	8.60%	16.3	-22
140	4.80E-05	0.40%	17.6	-0.9
130	1.50E-05	0.30%	113.1	-0.4
120	1.80E-05	0.42%	18.0	-0.5
110	2.20E-05	0.55%	17.9	-0.6
100	1.20E-04	2.10%	17.1	-1.4
97	4.50E-05	6.10%	17.6	-0.9
95	1.00E-03	13.00%	15.8	-2.7
93	4.50E-03	28.00%	14.6	-3.9
90	1.50E-01	89.00%	9.7	-8.8
80	5.00E-01	100.00%	4.9	-13.6

Figure 3



The -10 dB C/I was chosen partly because it was the maximum setting of the test equipment and partly because -10 dB is "reasonable" amount of energy to expect the modem to tolerate from an adjacent channel interferer.

figure 3 shows a plot of the degradation as a

function of the interfering carrier offset. From this plot it is apparent that the DSY modem experiences degradation due to an interfering source up to 190 kHz away. However in most cases a 100 kHz carrier spacing would be sufficient. Because of the burst nature of both packet and voice transmission systems the probability that both the interfering carrier and the wanted carrier are on at the same time is fairly remote, and a degradation of 3 dB from the nominal used in this test would likely result in a tolerable increase in packet retransmissions. If either the wanted carrier or the interfering source are likely to be on for long periods of time, or very low error rates are required, additional frequency separation of steps to reduce the level of interference should be taken.

4.2. Co-Frequency Interference

Table 2 shows the resulting $(E_b/N_0)_{eq}$, degradations, BER's and BLER's when the interfering carrier is co-frequency to the wanted carrier and the level of interference is varied.

Table 2
NBFM Co-Channel Interference

C/I (dB)	BER	BLER	E_b/N_{0eq}	Degr.	C/I_{eq}
8	3.70E-02	90.00%	12.3	-6.2	13.5
8	7.30E-03	75.00%	14.2	-4.3	16.2
10	1.70E-03	28.00%	15.4	-3.1	18.3
12	4.50E-04	7.90%	16.3	-2.2	20.3
14	1.60E-04	3.00%	16.9	-1.6	22.0
15	1.00E-04	1.90%	17.2	-1.3	23.1
17	4.50E-05	0.85%	17.6	-0.9	24.9
20	2.30E-05	0.40%	17.9	-0.6	26.8
25	9.10E-06	0.20%	18.3	-0.2	31.8

Figure 4

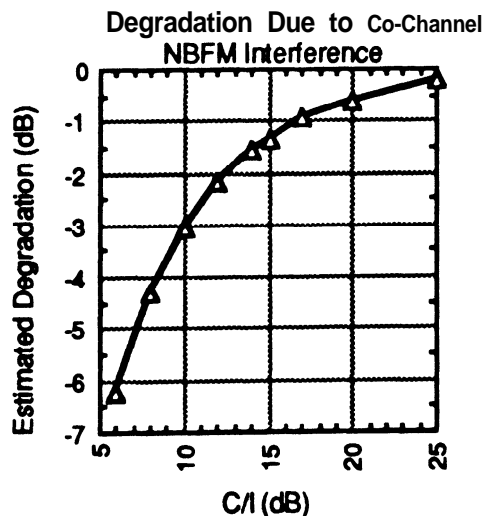


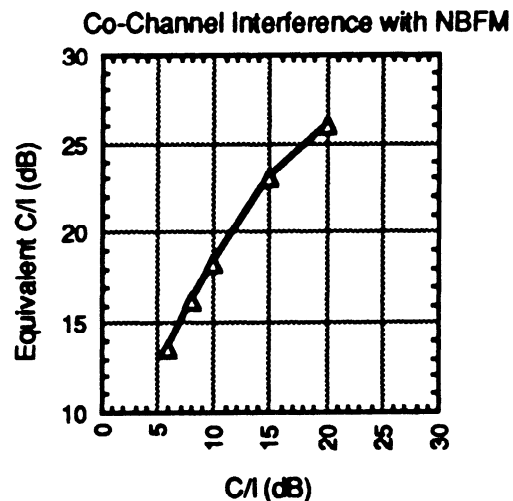
Figure 4 shows a plot of the degradation versus the C/I. The degradation curve is fairly well behaved. However, it is interesting to note that even a C/I of 25 dB causes a slight degradation in performance.

In order to calculate how NBFM will affect the DSY modem at other C/I's and other E_b/N_0 's a means of calculating the "equivalent" CA (C/I_{eq}), of the equivalent noise contribution of the interference, from the actual C/I is required. If the C/I_{eq} is known then it can be added to link budget calculations as another noise source. The C/I_{eq} is determined by calculating the increase in noise required to take the modem from the nominal performance to the BER with interference (the equivalent E_b/N_0).

$$C/I_{eq} = 10 \cdot \log \left[1 / \left(10^{-(E_b/N_0)_{eq} / 10} - 10^{-(E_b/N_0) / 10} \right) \right]$$

The graph of Figure 5 shows that C/I vs C/I_{eq} is

Figure 5



fairly linear. Closer inspection reveals that C/I_{eq} is approximately equal to C/I plus 8.0 dB. This is not quite what is expected. Although the ratio of the bandwidth of the NBFM to the wide band data is in the order of 8 dB, the interference in the band of the receive signal should affect the entire signal. That is the C/I and C/I_{eq} should be equal. However it appears that the demodulator is quite resistant to narrow band interference.

5. Performance With interfering Wide Band Digital Carriers

5.1. Adjacent Channel Interference

The wide band digital interference test was performed similar to the NBFM case. However, additional C/I's were tested. C/I's Of -3.0 and 0.0 dB were tested to simulate cases of:

- the 0 dB C/I case simulates the outputs from a single site with multi-DSY modem carriers being transmitted with equal levels;

- the -3 dB C/I case simulates the case where adjacent channel input carriers from users are set up so that virtually equal power levels would be received by a multi-channel repeater site;

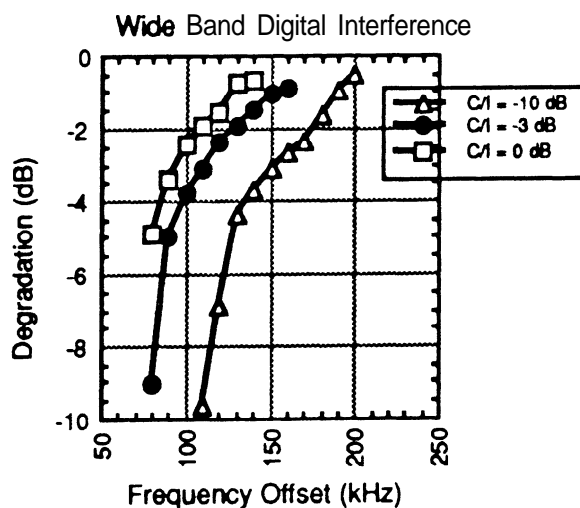
- the -10 dB C/I case simulates non co-ordinated network with users on adjacent channels transmitting vastly different power levels.

Table 3
Wide Band Data Adjacent Channel Interference with C/I=-10 dB

Frq. (kHz)	BER	BLER	$(E_b/N_o)_{eq}$ (dB-Hz)	Degradation (dB)
200	1.80E-05	0.42%	18.0	-0.5
190	5.00E-05	1.10%	17.5	-1.0
180	1.70E-04	3.70%	16.9	-1.6
170	5.90E-04	11.00%	16.2	-2.3
160	1.00E-03	17.00%	15.8	-2.7
150	1.80E-03	23.00%	15.4	-3.1
140	3.66E-03	33.00%	14.0	-3.7
130	7.70E-03	68.00%	14.1	-4.4
120	5.70E-02	99.00%	11.6	-6.9
110	2.00E-01	100.00%	8.9	-9.6

Figure 6 shows the plot of frequency versus

Figure 6



degradation. It is clear that in cases where there will be a large difference in received power for closely spaced carriers, that at least 150 kHz spacing should be used. It may be good frequency planning to space the carriers 200 kHz, thus adjacent networks could use the "in between" frequencies without causing interference.

5.2. Co-Channel Wide Band Interference

The final test is with the PSK carrier co-frequency to the DSY modem frequency. The results are given in Table 4.

Table 4
Wide Band Data Co-Channel Interference

C/I (dB)	BER	BLER	$(E_b/N_o)_{eq}$ (dB-Hz)	Degradation (dB)
2	5.00E-01	100.00%	4.9	-13.6
4	1.30E-01	100.00%	10.0	-6.5
6	2.90E-02	99.00%	12.6	-5.9
8	6.90E-03	72.00%	14.2	-4.3
10	1.80E-03	30.00%	15.4	-3.1
12	5.00E-04	10.00%	16.3	-2.2
15	1.20E-04	2.40%	17.1	-1.4
18	5.80E-05	1.30%	17.4	-1.1
20	3.00E-05	0.64%	17.0	-0.7
25	1.90E-05	0.42%	18.0	-0.5

Figure 7

Degradation Due to a Co-Channel PSK Carrier

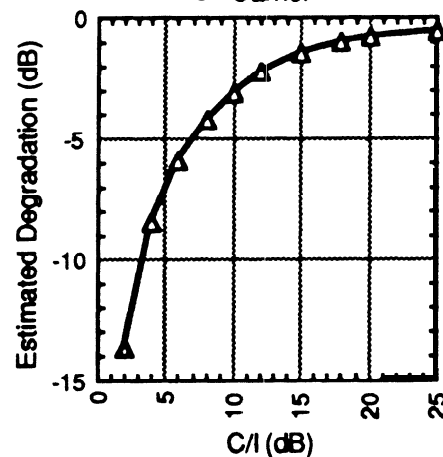
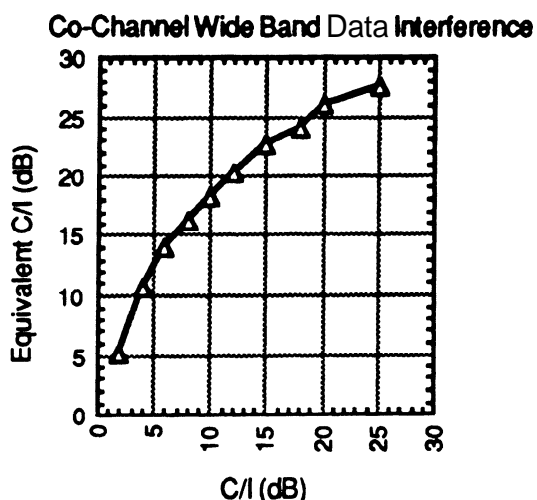


Figure 7 shows the degradation due to the co-channel PSK carrier. In order to achieve a degradation of less than 3 dB there must be at least 10 dB of isolation between networks. However this figure does not tell the whole story.

The result is fairly smooth and it would be expected that the relationship of C/I to C/I_{eq} would be linear and one to one. However as seen in Figure 8 the C/I_{eq} is better than the C/I by up to 8 dB. This is unexpected. Because a wide band noise like signal is placed co-frequency to the data signal it is expected that the degradation would be

Figure 8



dB for dB. However, it may be explained that:

- 1) the MSK receive characteristics have fairly significant sidelobes (as seen at 150 kHz offset in Figure 3),
- 2) the PSK interference spectrum does not have sidelobes that interfere with the MSK sidelobes,
- 3) the MSK modem may be receiving a significant amount of information from the sidelobes,

thus allowing better performance than expected.

For use in coordinating networks it is recommended that interference allocations be based on the actual C/I with corrections for the difference in occupied bandwidths. Using this allocation there would be some margin. Note, however, if the IF filtering is improved for adjacent channel rejection, co-channel interference performance would most likely suffer (this would be an interesting topic of future tests).

6. Recommendations and Conclusions

The results presented in this paper will be useful

tools for wide band digital carrier users in coordinating networks with:

- other wide band digital networks;
- NBFM packet networks; and,
- NBFM repeaters.

It is recommended that general coverage repeaters sites using multiple standard DSY modems should maintain a channel spacing of 150 kHz or more. Actual degradations or noise contributions due to interference may be taken from Figure 6 and Table 3. However, by modifying the receive filtering of the modem it may be possible to reduce the adjacent channel interference so that 100 kHz spacing may be used.

It is recommended that networks using DSY modems should maintain a minimum separation of 100 kHz to the nearest NBFM carrier. Again actual degradations and noise contributions can be taken from Figure 2 and Table 1.

For frequency reuse and co-frequency interference, the amount of isolation required should be based on the traffic mix:

- for common user repeaters and cases where high reliability is desired: link budgets should be calculated with a noise allocation for interference, if the interference is NBFM the C/I allocation should be 8 dB better than the actual C/I , if the interference is wide band data of another DSY modem carrier a noise allocation equivalent to the C/I should be added;
- for applications where high reliability is not required or where the probability of Carrier and Interference being on at the same time is small: the C/I should be at least 10 dB.

References:

- [1] Heatherington, D., "A 56 kilobaud RF modem", Sixth ARRL Computer Networking Conference, Redondo Beach, CA, August 27, 1987, pp. 68-75.
- [2] McLamon, B., "The 56 kb/s Modem as a Network Building Block: Some Design Considerations", Tenth ARRL Computer Networking Conference

THE 56 KB/S MODEM AS A NETWORK BUILDING BLOCK: SOME DESIGN CONSIDERATIONS

Barry McLarnon VE3JF
2696 Regina St.
Ottawa, ON, Canada K2B 6Y1
Email: barry@dgbt.doc.ca

ABSTRACT

In response to the now-famous cry of "Where is my high-speed RF?" [1], Dale Heatherington **WA4DSY** introduced his 56 kb/s modem design [2] to the packet community in 1987. Four years later, this breakthrough development has yet to make the major impact on the amateur packet networking world that many of us anticipated when it was introduced. We begin by examining some of the reasons why this is so, and then describe several of the possibilities for deploying the modem in the amateur packet radio network. It is the author's opinion that this modem is a versatile building block which can serve a **multitude** of roles in the network; if this article helps to stimulate more interest in pursuing them, then it will have served its purpose.

Introduction

There are a variety of reasons why the 56 kb/s modem (hereafter referred to as the **DSY** modem) failed to take off as quickly as we had hoped. Probably the major one was the lack of packet switch hardware which could make full use of the modem's speed. TNC hardware could not keep up, nor were there, until recently, any inexpensive HDLC interface boards for the PC bus available which could handle 56 kb/s without usurping all of the CPU's resources. This problem was resolved in 1990, with the arrival of the Gracilis **PackeTen** packet switch [3] and the Ottawa PI I/O card on the scene,

Other impediments to the widespread acceptance and deployment of the modem include the fact that the modem is only available in kit form, and that it costs significantly more than low-speed modems and TNC hardware. Some people do not appreciate the fact that this modem is a great deal more than a low-speed FSK modem scaled up to run faster; others are apprehensive that it will be difficult to construct and tune up. In fact, the modem is easy to build, and tune-up is straightforward; the most time-consuming and tedious part of getting one on the air is mounting the boards in a box and providing the requisite interconnections and +/- 5V power supplies.

Another problem area was, and continues to be, a shortage of 'affordable RF converters for translating the modem's input/output at 28-30 MHz to a suitable frequency above 220 MHz. In 1987, the choice boiled down to either a Microwave Modules **MMT220/28S** transverter (if the 220 MHz band was available to you at all), or the same firm's **MMT432/28S** model. Since then, the company has turned away from the amateur market, although the transverters can still be found quite often on the used market. Sinclabs of Toronto is now producing a 220/28 unit, and there may be a new source of 432/28 MHz transverters soon. Those currently available, such as from SSB Electronics, are very expensive. Transverters between 28 MHz and bands above 450 MHz seem to be nonexistent (transverters for the higher bands normally use 144 MHz as the IF).

If the modem is used for full duplex or split (half duplex, but with different receive and transmit frequencies) operation, then separate receive and transmit converters are needed, and this opens up a few new possibilities. There are several sources of receive converters available, and at least one source (Hamtronics) of transmit converters. Above 450 MHz the same situation as for transverters exists. With separate converters, however, it should be more feasible to run converters in cascade or put together custom

converters for the higher bands. The schemes outlined below, which would allow sharing of the converters by multiple modems, make it more acceptable to put some effort into assembling the necessary RF hardware.

The 56 kb/s CSMA MAN

In Ottawa, 56 kb/s packet began in mid-1988, with a link between VE3JF and VE3MDL on 220.55 MHz. We were soon joined by VE3OCU, and I found that I could not hit both stations reliably with a single beam heading. The idea of putting up a 56 kb/s repeater to support our new MAN (Metropolitan Area Network) was born soon thereafter! We decided that anything less than CSMA with no hidden transmitters would be unacceptable, so we began assembling a full-duplex repeater. Band plan limitations and the lack of a 220 MHz duplexer seemed to preclude doing an in-band repeater, so it was decided to make the repeater cross-band, with input on 220.55 MHz and output on 433.55 MHz. At our home stations, this meant the addition of a new modem receive crystal, a 432/28 MHz receive converter, and a 432 MHz antenna.

The repeater (see Figure 1) is a bit regenerator, similar to the arrangement which some people have used at lower bit rates: the demodulated data from the receive side becomes the transmit data stream, and the transmitter is keyed by the demodulator's DCD output. In order to minimize the keyup time, we keep the transmit side of the modem running continuously (ETS asserted) and key only the transmit converter. One word of warning about this practice: the transmit clock is derived from a 3.579 MHz oscillator on the encoder board, and the third harmonic of this signal falls within the passband of the 10.7 MHz first IF in the demodulator. Make sure there is no interference problem by observing the eye pattern signal while keying the transmit side on and off via the RTS line. Under no-signal conditions you will see a reduction of the noise amplitude at this point if the interference is present. In my modem, the interference was noticeable only with the cover of the box completely off, but of course your modem may be packaged quite differently from mine (I prefer to arrange the boards in the same horizontal plane, with quite a bit of spacing between them).

There is one complication with using the DSY modem as a bit regenerator: the transmit encoder insists on clocking in the data using its internal clock; however, the timing of the data is determined by the transmit clock at the station which is being received at the repeater input. These clocks are not phaselocked, so the resulting skew can cause errors in sampling the data and failure to repeat it faithfully. I designed a simple 32-bit FIFO (First-In, First-Out) buffer circuit to correct for the clock skew, along with a watchdog timer and some logic that would allow a switch to be connected at the repeater site.

Except for some antenna problems (we found that some of those nice-looking fiberglass gain omnis from the Far East don't stand up to Canadian winters in exposed locations too well!), the repeater has been very reliable since it was installed in January 1990, on top of a building about 26 stories in height. Thus far the furthest users are about 20 km from the repeater, and we have found that 10 Watts output is perfectly adequate. Judging from the link margins, the effective coverage is probably at least two or three times this distance. The users run from 0.5 to 10 Watts, and except for those close to the repeater, most use small yagis for both receiving and transmitting. Connection of the repeater to our packet switch at the site was completed in the Fall of 1990. The switch is a 12 MHz PC AT-class machine running KA9Q NOS. Aside from the 56 kb/s MAN port interfaced via a PI board the switch has a couple of ports which provide access to/from the low-speed NET/ROM network and an ethernet board which provides an Internet mail gateway and access to future Unix-based servers. We expect to further upgrade the capabilities of the switch soon, with the addition of a PackeTen board. More information on the repeater was published in the New England TCPer [4]. Copies of the article (and information on the PI board) are available from the author. For information on the WA4DSY modem itself, contact GRAPES, P.O. Box 871, Alpharetta, GA 30239-0871, USA.

Before leaving the topic of setting up a 56 kb/s MAN, it is worth noting that anyone setting up a 56 kb/s network of any sort should be aware of the need for adding front-end filtering to the receive side of the stations. The receive converters have very broad front ends, and strong signals several MHz away can wreak havoc with your reception. So, while you're thinking about how to set up your local network, be on the

lookout for surplus **bandpass** cavity filters and the like. The modem's 28 MHz RF input is also very **broad**, and improvement at that point will sometimes suffice instead of front-end filtering. More about that later.

The **Cellnode/Feeder** System: A New Role for 56 kb/s

The current evolutionary trend in the packet network is towards having a collection of major **multiport** nodes ("node stacks"). Typically, a node has one or more ports for users to access the network, and one or more ports for links to other nodes ("**backbone**" links). In cable TV parlance (not a great analogy, but the terminology is useful here), we could call the node-to-user links "drops" and the node-to-node backbone links "trunks". Nearly all of the current drops run at 1200 b/s, whereas most of the trunks operate in the 1200-9600 b/s range, although there are a few running at 38.4 kb/s and 56 kb/s (and some higher-speed links are under development, notably in California). The nodes are generally at good RF sites, and because such sites are not easy to come by, they are usually spaced fairly far apart. This arrangement works pretty well in areas with relatively low user population densities, since only a small number of users needs access to a given node. In an urban area, however, the node may need to **serve** a large number of users. A single user port quickly becomes saturated, and the users get frustrated with the resulting poor throughput. The collapse of the user port from congestion is hastened by the hidden transmitter problem. The obvious solution is to add more user ports, but this may not be feasible. Maintaining isolation between the different RF ports of the node, and probably with other radios at the same site, becomes increasingly difficult as new frequencies are added. The hidden transmitter problem can be eliminated by putting up a full-duplex repeater for the user port, but this also involves complications on the RF side, and this solution has not been a popular one.

Now let's back up one step and take a look at the problem from the point of view of user access to a single node, leaving aside the question of inter-node links for the moment. Let's assume that the users are running low speed (1200-9600 b/s), so that only a small number can share a channel if they are to get adequate

throughput. Hidden transmitters should be avoided in order to contribute to this objective. On the other hand, we want to keep things simple at the node site; **ideally**, we would like to have just a single half-duplex port for users. How do we simultaneously meet these requirements? The answer is quite obvious: we must restrict the coverage of the node so that it covers a relatively small geographic area. Then the number of users accessing the node will be small, and being close together, they likely will be in range of one another and will not be hidden transmitters. And, since coverage is quite limited, the frequency can be reused with smaller geographic spacing. In other words, we adopt a cellular approach in setting up our network.

Having limited-coverage cellular nodes (for convenience, I will call them cellnodes) sounds like a fine idea for giving users better access to the network, but we're not out of the woods yet. How do these cellnodes get tied into the network? With their limited coverage, they may not have good paths to their neighbors, not to mention the more distant nodes outside the urban area. Clearly there is still a need for well-sited central node which can provide the trunks to the more distant areas and also tie the cellnodes into the network. Again the similarity to the cable TV network: in addition to the trunks connecting the major nodes, and the drops from the cellnodes to individual users, we now require a feeder system which ties a group of cellnodes into a nearby major hub node. Now we come to the crux of the matter: how do we construct the feeder system? We could run separate point-to-point links, each on a different frequency, to each cellnode. This is okay from the point of view of the cellnodes, but it makes the design of the central node horrendously difficult. The point-to-point approach scales very poorly: each time we add another cellnode, we must add another port to the central node, the necessary radio and antenna hardware, and worst of all, the requisite filters and duplexers needed to eliminate interference between all the radio gear (and good sites tend to have other amateur and non-amateur radios present in addition to the packet equipment). At some point, it will become virtually impossible to add another cellnode.

Enter the CSMA MAN with full-duplex repeater. Provided that the bit rate is high enough to handle the combined traffic with minimal delays, a CSMA shared-channel arrangement will work fine for the feeder links, and it vastly

simplifies the setup at the central node. Only one set of radio gear and omni antenna (two antennas if it's a crossband repeater) is needed, and instead of a stack of TNCs or equivalent, a single port on some hardware that is more appropriate for doing the job of a packet switch. Not only is it simple, but it scales well: a new cellnode can be brought up without making any change whatever at the central node (except possibly programming in some new routing information), and the overall throughput of the feeder system degrades gracefully as new cellnodes are added. Of course, at some point the delays will become noticeable and adding more cellnodes will be unacceptable. On a 56 kb/s full-duplex feeder system, it will take a *lot* of cellnodes to reach that point if each has just a handful of 1200 b/s users, but it must be assumed that at least some of the cellnodes will be providing drops at up to 9600 b/s. Here's where it gets interesting.

Naturally, my candidate for building the feeder system around is the DSY 56 kb/s modem. As you will recall, this is an RF modem which provides its input and output separately at frequencies in the 28-30 MHz band, and transmit and receive converters are used to translate this IF to/from the actual operating frequency (or frequencies, in this case). This property is a big win when it comes to expanding the capabilities of a node. Consider Figure 2, in which we have added a second full-duplex repeater at the node site. Adding another repeater sounds like a formidable proposition, but take a look. New antennas, filters duplexers, radios (converters)? There aren't any! All we need is the new modem and its associated FIFO and interface hardware, a power splitter at the 28 MHz output of the receive converter, and a power combiner at the 28 MHz input to the transmit converter. And, of course, the packet switch has to have another high-speed port available - not a problem if we've chosen hardware appropriate for doing the job of a major network node, such as the PackeTen board. That's it - we have doubled the capacity of the feeder system, with no sweat.

Okay, I've glossed over some details - life is never that simple! Let's consider some of the finer points... First of all, it's quite obvious that we must set the drive levels into the transmit up-converter such that each repeater only uses half of the total power output. In fact, the drive should be backed off a bit more than that in order to be certain that the converter/amplifier remain in their linear operating regions. It would be wise to use

a spectrum analyzer at the RF output to make this adjustment. Chances are that link margins will be sufficient on the links using the original repeater so that the 3 dB loss in repeater power output won't cause any problems, but if not, this is easily remedied by adding a suitable "brick" linear power amplifier at the repeater output. Since the repeater is full-duplex, transmit/receive switching is not needed, provided that the amplifier is stable when its drive is removed.

There are also some constraints on the operating frequencies of the repeaters. The receive and transmit converters themselves don't impose much restriction, since they are inherently broadband devices. Clearly the repeater inputs must be in the same band (within 2 MHz or so), and the same goes for the outputs. The more severe constraint comes from the need to place both inputs within the passband of the input bandpass filter, preferably without modifying the filter. There may also be narrowband filtering at the combined repeater output, if the repeaters are in-band or duplexed with other radio equipment. This means that the repeater inputs and outputs should be within a few hundred kHz of each other (in the case of 56 kb/s, probably occupying adjacent, or perhaps next-to-adjacent, 100 kHz channels). In some areas, several contiguous 100 kHz bandwidth channels for packet may not be available on the band(s) of choice; for example, the recent band plan for the 222-225 MHz band in Southern California precludes this type of usage. Hopefully, the advantages of block conversion and adjacency of wideband packet channels will be given more consideration in future band planning.

Assuming that are adjacent 100 kHz channels available, we now must consider whether the DSY modem is up to the task of operating in such a mode. The modem is a double-conversion device: the 28 MHz front end is converted to a first IF of 10.7 MHz, and then to a second IF 455 kHz. The front end is very broad; in fact, it only has a lowpass filter to bandlimit the input. This limitation was brought home to us in Ottawa when several of us were experiencing interference problems when receiving the output of our repeater on 433.55 MHz. The interference was eventually traced to a surprising source which was well away from that frequency: a network of paging transmitters operating near 414.7 MHz were being down-converted in our receive converters to about 10.7 MHz, and the signals were sailing right through the modem front end and

reaching the demodulator. This prompted Dennis Rosenauer (VE7BPE) to design a 28-30 MHz bandpass filter to go between the receive converter and the modem (details available from the author). The first IF of the modem has a bandwidth of about 300 kHz, and the second IF establishes the ultimate bandwidth of the modem (when used at 56 kb/s) of about 70 kHz. This means that there will be very little rejection of an adjacent-channel signal 100 kHz away until we get to the second IF, and thus there is potential for intermod problems in the earlier stages of the modem if the adjacent channel signal is much stronger than the desired signal. Furthermore, the second IF filter consists of only three relatively low-Q LC fitters, and it rolls off quite slowly, so the ultimate adjacent-channel rejection is not very impressive. Recent measurements by Ian McEachern VE3PFH [5] indicate that the adjacent-channel rejection is insufficient to avoid significant degradation of the bit error rate, even under the most optimistic scenario (we control the power of the stations using the repeaters such that signals in the adjacent channel never exceed the desired signal by more than 3 dB). We also have to be concerned about the possibility of interference from other types of emissions on nearby frequencies.

Tightening up the first IF of the modem is quite possible, but far from trivial. Simply replacing the 10.7 MHz ceramic filter, which is specified at about 280 kHz bandwidth, with one having 80-100 kHz bandwidth is problematic since the latter is not generally available. Filters for 10.7 MHz bandwidth generally come in two types: those which are intended for FM stereo broadcast use, which requires a bandwidth of around 250 kHz, and those intended for narrowband FM, with a bandwidth usually less than 20 kHz. Improving the skirts of the 455 kHz second IF filter is another alternative, but again does not lend itself to off-the-shelf solutions. We are continuing to study the possibilities for upgrading of the modem's adjacent channel interference rejection capabilities; in the meantime, a stopgap solution is to use next-to-adjacent channels (i.e. 200 kHz spacing).

As for the power splitter and combiner, they need not be anything elaborate. The splitter may not even be needed. I have frequently connected my HF receiver to the output of the receive converter in parallel with my DSY modem by just using a coaxial tee connector, with no apparent ill effects on the modem performance (this is a

good way of checking for interference problems, by the way). It is probably more important to use a true hybrid circuit, with reasonable isolation between the inputs, for the power combiner. Such units are available from companies such as Mini-Circuits, but they are not difficult to build. Check the "Test and Measurements" chapter of a recent ARRL Radio Amateur's Handbook for construction information.

For obvious reasons, in the two-channel repeater configuration we cannot let the transmit side of both modems run continuously and just key the transmit converter, as was recommended for the single repeater. The DSY modulators must be keyed, but the present method of keying could be changed in order to improve the keyup time.

In Ottawa, our 56 kb/s full-duplex repeater was originally intended as a MAN for the TCP/IP "power users". It still fulfils that function, but it has also evolved into a feeder system for several BBS stations and 1200 b/s user access ports. We are now working on a second repeater, to be deployed as outlined here, which will be used to separate those functions to some extent. We also are investigating the possibility of running the second repeater at a higher speed, such as 112 or 128 kb/s. Despite those objectives, the mixed usage of our current MAN has worked out quite well. The power users can become part of the network feeder system by simply opening up a low-speed port and thereby creating a new cellnode for their local area. Of course, highly motivated users like this are not all that common, so it might be necessary for a local club to subsidize some of the cost of putting up a cellnode. Either way, the cellnode sites will generally be home stations (running KA9Q NOS), which can be a major advantage in terms of network maintenance and reliability. It is unlikely that the coverage provided by cellnodes in a given area will be complete, so it will probably be necessary to retain a wider-coverage low-speed port at the main node site to pick up the users who are not covered otherwise.

A Fly in the Ointment: Multipath

One disadvantage of the CSMA MAN compared with separate point-to-point links is that multipath problems are more likely with the former. This is because the antenna at the central node must of necessity have omnidirectional

coverage, and therefore it does not provide any discrimination against signals arriving (or leaving) by paths other than the direct one. That's not to say that such problems will happen frequently, but the possibility should be kept in mind when setting up links,

Multipath is characterized by a delay spread parameter, which gives a measure of the time interval over which significant energy from a given signal element from a transmitting site arrives at a receiving site. As far as data transmission is concerned, the multipath has quite different manifestations, depending upon the relationship between the delay spread and the symbol length (baud interval) being used. If the delay spread is much smaller than the symbol length, then the multipath can cause more or less complete cancellation of the signal (i.e., "flat" fading, where in this case the fade is permanent, at least until something in the environment between the sites changes). This can happen, for example, if you are receiving the signal on the direct path, and in addition are receiving it with nearly equal strength after it reflects from a nearby building. If the signal level on a link proves to be much lower than expected, it is worthwhile trying to adjust the position of the antenna at the cellnode end of the link to see if that improves the situation (with the DSY modem, looking at the eye pattern is the best way to assess the link quality). If it does improve, hopefully the paths will be stable enough to retain the favorable phase relationships needed to avoid cancellation.

The other situation occurs when the multipath spread is an appreciable fraction of the symbol length, say 20% or more. Then we do not experience complete signal cancellation, but we have a new problem: intersymbol interference. A significant amount of energy from the preceding symbol is still arriving while we are trying to decide the value of the current symbol, making it harder to make the correct decision. The bit error rate goes up rapidly as the intersymbol interference increases, and since every bit in a packet must be correct for the packet to be successfully decoded, link performance falls off very rapidly. In the case of a 56 kb/s binary modem like the GSY, the symbol length is about 18 μ s, so problems from intersymbol interference start to get likely when the delay spread exceeds 2 or 3 μ s. Unfortunately, spreads of this magnitude and much more are not at all uncommon, especially in urban and mountainous areas. The author was involved in a series of multipath delay spread

measurements (at 800 MHz) in four Canadian cities in 1990, and we frequently saw spreads in excess of 10 μ s, and more than 20 μ s in several cases. Other workers have documented multipath spreads at VHF of as much as 50 μ s in some unusual situations.

The insidious thing about multipath-induced intersymbol interference is that, unlike the case of signal cancellation mentioned above, a small shift in antenna location probably won't help at all. The best way to avoid multipath is to move to the prairies, but a more practical solution for most people is to use directional antennas. Many of the long-delayed echoes arriving at a site will have quite a different azimuth from the direct path, and a gain antenna will often attenuate these enough to solve the problem. In the case of the CSMA MAN, this means that all of the stations except the central node should use directional antennas (this may also be advantageous in terms of keeping signals on the MAN repeater input frequency confined to a smaller area, permitting greater frequency reuse). Since the central node must have omnidirectional (or at least very wide) coverage, multipath will generally be more of a problem than on point-to-point links. For this reason, we probably will not see CSMA MANs running at speeds much higher than 56 kb/s in the near future. Intersymbol interference can also be dealt with by using tapped delay-line equalizers, but doing so in a CSMA MAN environment would be problematic, to say the least.

A useful tool for assessing link quality is the demodulator eye pattern as observed with an oscilloscope synchronized to the recovered clock. These signals are readily available in the DSY modem. Packets flash by pretty quickly at 56 kb/s, so some provision should be made for keying up the modem/transmitters for longer periods for link quality testing. A very noisy eye may indicate a cancellation problem caused by multipath, or it may be some other fault which is lowering the link margin. An eye which is badly distorted rather than noisy may be a good indication of multipath intersymbol interference. If the latter is noticeable even when a yagi or other directional antenna is used at the cellnode site it is worthwhile rotating the antenna to see if the eye signal improves. Minimum distortion may not coincide with the antenna being aimed along the direct path.

Building Major Node Sites

The concept of block conversion (sharing of RF components) can easily be extended to the network backbone links, or trunks. Although the major trunks in the network will eventually migrate up to microwave links running at T1 speeds (1.544 Mb/s) and beyond, 56 kb/s will satisfy much of our needs in the near term. If we're going to do the job right, though, we have to make the trunks full duplex. The reasons are several: obviously, higher efficiency and better throughput due to the simultaneous two-way data transfer (provided that the hardware and protocols support it) and elimination of turnaround time overhead. In the case of the DSY modem, the noncoherent demodulator could be replaced with a coherent design, thus improving link margins. The really big win with full duplex, however, is the ability to use the block conversion and multicoupling scheme outlined above for multichannel MAN installations.

Consider a node with three 56 kb/s trunk links. The current tendency would be to put up three half-duplex links, each on a different band, with a complete set of radio gear and directional antenna for each one. This may not be feasible at some sites. On the other hand, if we use a variation on the block conversion scheme outlined previously, we can run full duplex on all the links and use only two bands, with just one shared receiver on one band, and a shared transmitter on the other. In fact, all three full-duplex links could be done on a single band, using a duplexer and one antenna; however, current band plans are not conducive to this type of operation with high-speed packet. The antenna(s) can either be omnidirectional with gain in the vertical plane, or an array of yagis or other directional antennas with a suitable matching harness. As in the multiple repeater idea discussed above, the poor adjacent channel rejection capabilities of the modem are a limiting factor in this full-duplex multiport scheme. Greater than 100 kHz spacing between channels is needed unless the modem IF response is improved.

A further simplification is possible: all of the links could share a single modem and frequency slot on the transmit side. This configuration was proposed by Phil Karn several years ago [6]. Unfortunately, the channel access protocols currently in use would not make efficient use of this arrangement. The ports of the node would have to share the transmit channel on a first come first

serve basis, and the links would have to be operated in half-duplex mode to permit this (though it seems to me that some minor software tweaks could produce a hybrid "full duplex when the transmit channel is available" mode). Such a configuration is worth considering in cases where 100 kHz bandwidth channels are in short supply.

For further information on engineering full-duplex inter-node links at 56 kb/s, I recommend reading Don Lemke's article in the 1988 ARRL CNC proceedings [7]. His article also has a good discussion on the calculation of link margins.

References

- [1] Fox, T., "RF, RF, where is my high speed RF?", *Fifth ARRL Computer Networking Conference*, pp. 5.1-5.5.
- [2] Heatherington, D., "A 56 kilobaud RF modem", *Sixth ARRL Computer Networking Conference*, Redondo Beach, CA, August 29, 1987, pp. 68-75.
- [3] D. Lemley and M. Heath, "The Packet-Ten system - the next generation packet switch", *Ninth Computer Networking Conference*, London, Ontario, September 22, 1990, pp. 170-176.
- [4] McLarnon, B., "A packet repeater using the WA4DSY 56 kbs modem", *The New England TCPer*, Vol. 2, No. 5, May/June 1990.
- [5] McEachern, I., "Digital networking with the WA4DSY modem - adjacent channel and co-channel frequency reuse considerations", this conference.
- [6] Karn, P., "A, high performance, collision-free packet radio network", *Sixth ARRL Computer Networking Conference*, Redondo Beach, CA, August 29, 1987, pp. 86-89.
- [7] Lemke, D., "Cellular area coverage transport networks", *Seventh ARRL Computer Networking Conference*, Columbia, MD, October 1, 1988, pp. 122-134.

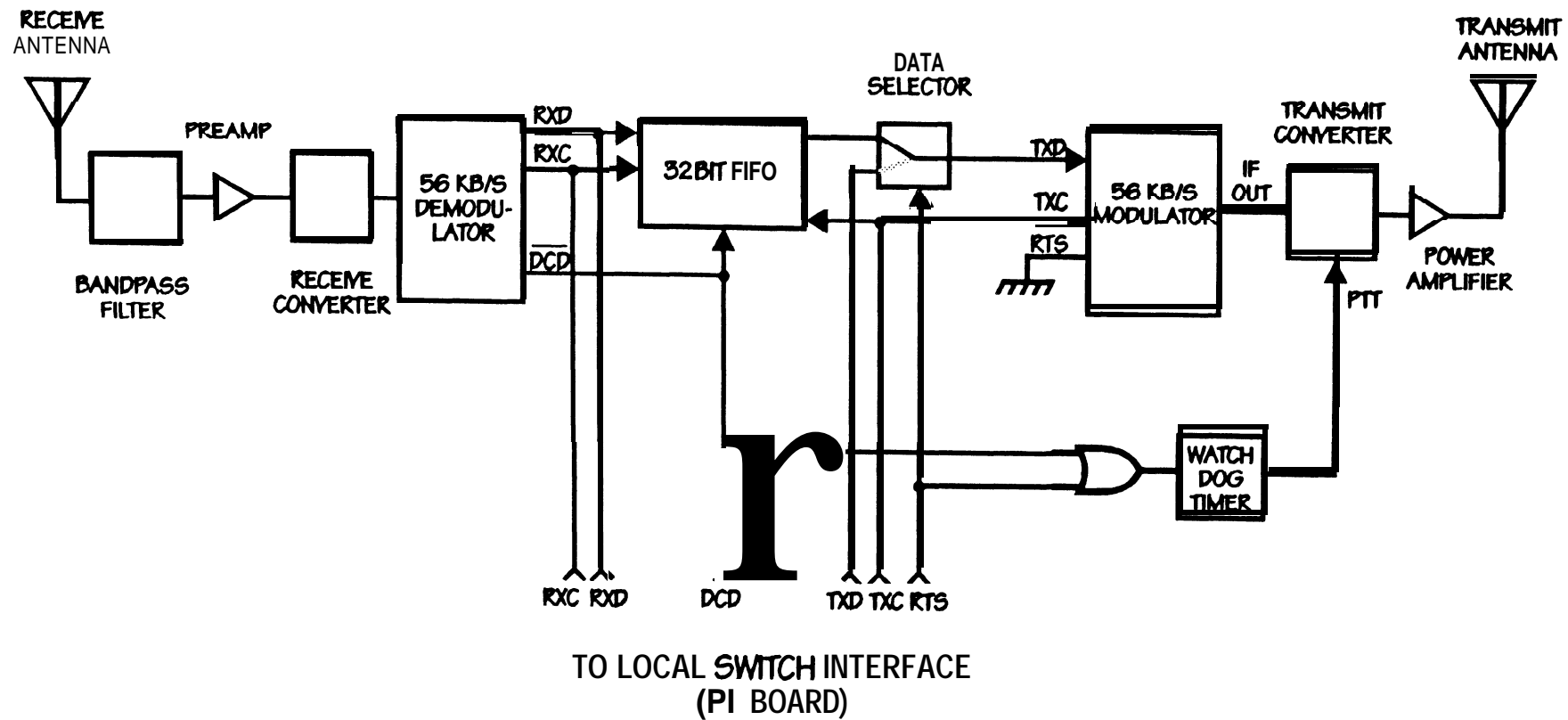


Figure 1 56 KB/S REPEATER BLOCK DIAGRAM

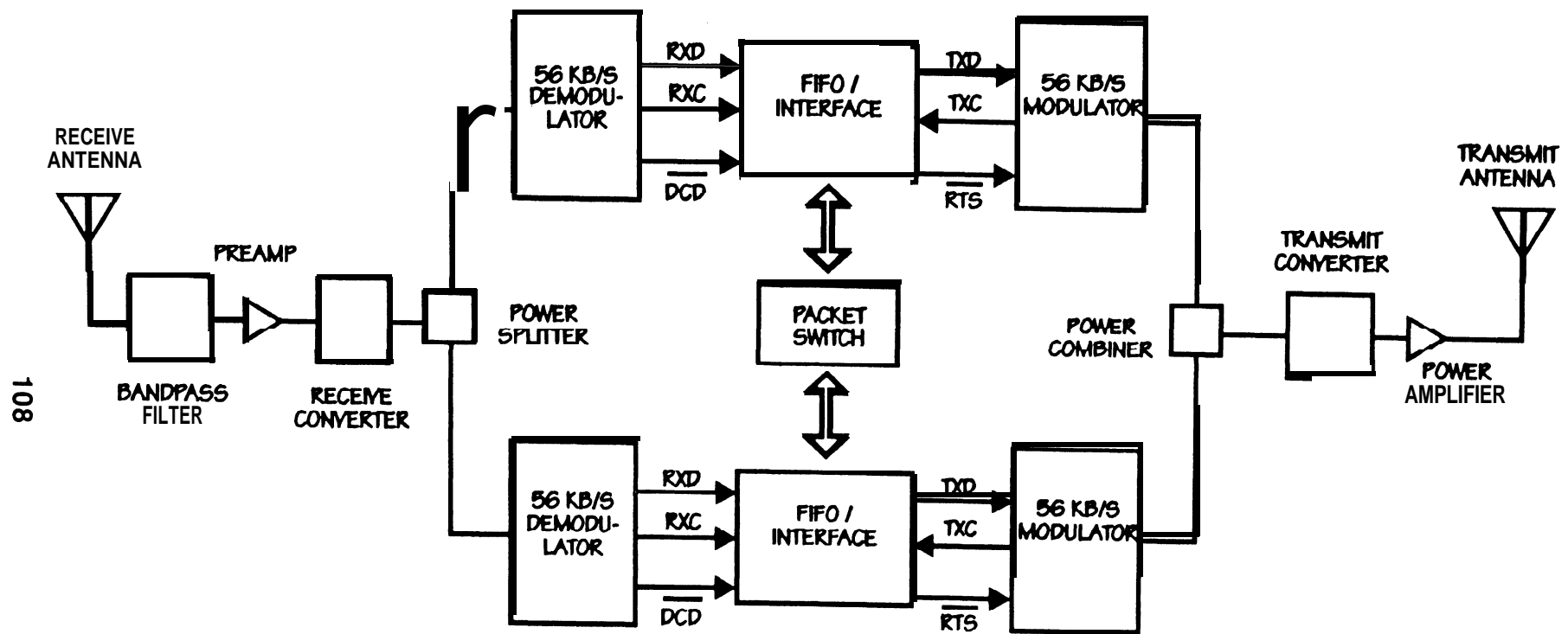


Figure 2 TWO-CHANNEL REPEATER CONFIGURATION

Multi-Drop KISS operation

by Karl Medcalf WK5M

John Wiseman (G8BPQ) has developed a multi-drop KISS protocol, allowing several TNCs to be connected to a single serial I/O port while operating KISS mode.

To prevent all TNCs from responding to all data from the serial I/O port (called the master), each TNC must have a specific address. This address is contained in the high nibble of the command byte in each KISS frame.

The KISS format is shown in Figure 1.

For multi-drop KISS, the Command byte is modified by changing the first digit of the HEX value listed in Figure 1. As listed, the TNC with address 0 would be addressed. For TNC address 1 the command bytes would be:

\$10	Data to be transmitted
\$11	Set TNC TXDELAY
\$12	Set TNC Persistence
\$13	Set TNC Slottime
\$14	Set TXtail
\$15	Set Full Duplex mode
\$16	Set Hardware
\$FF	Exit KISS mode

Note that since the \$FF code applies to all units, a single \$FF command byte would cause all TNCs on the serial I/O port to leave KISS mode.

Within any given frame of data, the FEND character can only appear as the opening and closing characters, thus delimiting the frame. If the \$CO value must be sent as data or as an argument to a TNC command (\$x1, \$x2, or \$x3 command byte), then standard KISS transparency must be applied.

The G8BPQ multi-drop KISS has extended the KISS command bytes listed above by adding some new values for the low-order nibble. These additions are:

\$xC	Data to be transmitted – Acknowledgement required
\$xE	Poll frame (see Polled Mode below)

Acknowledgement Mode

The G8BPQ Acknowledgement Mode is intended mainly for HF operation where data that has been sent to the TNC for

FIGURE 1

BYTE0	FEND	\$C0
BYTE 1	COMMAND BYTE	\$00 – data to be transmitted \$01 – set TNC TXDELAY \$02 – set TNC Persistence \$03 – set TNC Slottime \$04 – set TXtail time \$05 – Full Duplex mode \$06 – Set Hardware \$FF – Exit KISS mode
BYTE 2-N	Data	only if command byte is 0
BYTE2	Parameters	Only for non-zero COMMAND byte
BYTE N+1	FEND	\$C0

transmission may not actually be transmitted for an extended period of time. When this mode is selected, the master unit would send the special **\$xC** command byte, causing the selected **TNC** to return an acknowledgement when the **frame** has been sent. This allows the master to start his timing parameters (**FRACK**, etc.) at the time the frame actually left the TNC. For instance, if the master wants to send a data frame to TNC with address **5** and wants to use the Acknowledgement Mode, the frame would begin with:

\$CO \$5C \$aa \$bb data to be transmitted **\$CO**

The **\$aa** and **\$bb** in this example are not transmitted, but are returned to the master by the TNC when the **frame** is transmitted. This allows you to sequentially number the frames if you desire, as these two bytes may be any value. When the TNC has transmitted this frame it will send the following back to the master:

\$CO \$5C \$aa \$bb \$CO

Polled Mode

The **G8BPQ** extended **KISS** code also allows a Polled Mode. In this mode, none of the **TNCs** will send any data to the master until the master polls the specific TNC for data. The poll **from** the master consists of the following:

\$CO \$xE \$CO where the **x** is the address of the unit

If the addressed TNC has data, that data is sent to the master (only one frame is sent for each poll, even if more are available). If

the addressed **TNC** has no data, the Poll frame is returned to the master (**\$CO \$xE \$C0**).

Checksum Mode

One final mode now implemented in the **G8BPQ** extended **KISS** protocol is a Checksum Mode. In this mode, data sent between the master and the slave **TNCs** contains a checksum to insure that the data did not get corrupted on the serial I/O line. The checksum is a simple exclusive OR of all data between the opening and closing **FEND** characters. If this data contains any transparency processed data, the **REAL** value is used and not the escaped characters. The checksum is a single byte, and is sent before the closing **FEND**. Checksum may be used in conjunction with any or all of the above modes.

When several **TNCs** are connected to a common RS-232 serial I/O line, the data output from the master (**TXD** on a computer or **RXD** on the Data Engine) and the ground will be directly connected to the **TXD** line and ground of all slave **TNCs**. The **RXD** line from each of the **TNCs** must have a single diode connected with the **ANODE** to the slave TNC and the cathodes all tied together. The cathode side is then connected to the master data input (**RXD** on a computer or **TXD** on the Data Engine).

Since these modes are not directly compatible with “standard” **KISS**, the firmware must provide some method of selecting the various modes of operation. The **Kantronics** implementation of the extended **KISS** has added the commands shown in Figure 2 to select operation.

FIGURE 2

MYDROP	(0-15)	Specifies the TNC address
CHECKSUM	(ON/OFF)	Enables Checksum Mode
POLLED	(ON/OFF)	Enables Polled Mode
INTERFACE	(TERMINAL/KISS/BPQ)	TERMINAL allows setting parameters KISS implements “standard” KISS BPQ enables the extended BPQ KISS

John's implementation of this extended KISS protocol should enhance KISS type operations, especially on HF where timing becomes a particular problem. The ability to "multi-drop" many TNCs from one master (computer or Data Engine) allows multi-port networking nodes from a single serial port on a computer, thus reducing hardware requirements.

Software which may be downloaded from the Kantronics BBS at (913) 842-4678. John is preparing multi-drop KISS eprom images for other TNCs and will distribute them along with his standard PC based G8BPQ code. When these other multi-drop KISS eproms become available, they will be included in the Data Engine BPQ distribution.

KISS Frame

C0	Command Byte	Data
		\$FF Exit KISS mode
		\$06 Set Hardware
		\$05 Full Duplex mode
		\$04 set TXtail time
		\$03 set TNC Slottime
		\$02 set TNC Persistence
		\$01 set TNC TXDELAY
		\$00 data to be transmitted

G8BPQ Multi-Drop KISS Additions

0	0	0	0	1	1	0	0	\$OC	Data to be transmitted – Ack required
0	0	0	0	1	1	1	0	\$OE	Poll frame
0	0	0	0	x	x	x	x	\$0x	TNC address 0
0	0	0	1	x	x	x	x	\$1x	TNC address 1
1	1	1	1	x	x	x	x	\$Fx	TNC address 15

Where x =
any KISS
Command Byte
(as shown above)

Checksum Mode

			XOR Checksum	
--	--	--	-----------------	--

THE SHAPE OF BITS TO COME

James Miller BSc, G3RUH

3 Benny's Way
COTON
Cambridge
CB3 7PS
England

ABSTRACT

This tutorial article is about bits, about bandwidth and about control of both. Lately a number of expressions have crept into amateur radio data transmission, creating both interest and puzzlement. What, for example, is the "raised cosine modulation" used on microsats AO-16/18/19? And what is the RSM-8 on Rudak/AO-21? What PSK modulation does Fuji-20 use? Come to that, what is PSK or BPSK? Read on.

Let's come clean at the outset; "raised cosine", RSM-8, PSK and DPSK are the same thing. They're all forms of PSK - "phase shift keying".

In the amateur environment we are usually trying to send data as fast as possible within a limited RF bandwidth. This limit may be set by the receiver alone, or it may be due to some statutory, aesthetic or technical requirement. So a very important part of data transmission concerns the data's spectrum.

UNDERSTANDING PSK

Let's begin with a statement and explain later:

PSK spectral considerations at RF can be analysed by examining the characteristics of an isolated bit at "DC" or baseband.

I have to start somewhere, so I'll make the assumption that you have in your head an image that "data" is the sort of regular signal you have running down a piece of cable. On a 'scope it looks like random highs and lows spaced at regular intervals. Each of these elements is called a "bit", and they flow at the "bit rate", described

as 9600 bits/sec or sometimes "9600 baud" depending on the context. See the waveform "Data In" of fig. 1.

There are many ways to modulate this data onto an RF carrier, for example ON/OFF keying, carrier FSK, two-tone AFSK and so on. Here I am concerned with double sideband modulation, DSB. That is, the data signal drives a balanced modulator whose other input is the RF carrier. The output then is simply a frequency shifted replica of the input data. (Remember what happens when the input is voice?)

Now binary data has only TWO states. It is either a "1", represented by +1 volt say, or a "0", represented by -1 volt. And that means that the modulated RF carrier also

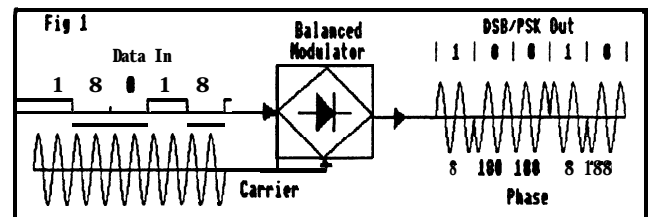


Fig 1. PSK is usually generated by a DSB balanced modulator. So DC spectrum is simply shifted to RF. Data "1"s give carrier phase 0°, data "0"s give phase 180°.

takes only TWO states; either it has phase 0 degrees or 180 degrees. This is because compared with the +1 volt level, the -1 volt inverts the carrier. That means a 180 degrees phase shift. Voila! Binary Phase-Shift-Keying or BPSK. The term PSK, often used, is actually imprecise as it embraces other shifts as well as 0/180 degrees. "Antipodal PSK" is correct but long-winded.

So, if we know the spectrum of the "baseband" data signal, then the RF spectrum is just its replica above and below the carrier frequency, because it's DSB, double sideband suppressed carrier.

If in addition, the data signal has amplitude variations, then these will translate into identical amplitude variations at RF.

This is a very important equivalence, and it's worth restating:

"Binary data DSB modulating an RF carrier" and "binary phase shift keying (BPSK)" are exactly the same thing. If we want to control or analyse the RF spectrum characteristics, we only need to control or analyse the source data characteristics.

UNDERSTANDING DATA STREAMS AND ISOLATED BITS

We now make a reasonable assumption. The data stream consists of random bits. If the data is random, for the purposes of analysis we don't then need to know anything about the content of the message stream. All we need to know is the properties of ONE ISOLATED BIT. Whatever properties we can discover about the isolated bit will also apply to the average summation of the random assortment of them that make up the stream.

so in summary, our problem of spectral considerations at RF can be analysed via the relatively simpler job of examining the characteristics of an isolated bit at source level, DC or "baseband".

As a communications theory aside, it is worth noting that with efficient communications, that data MUST in fact be

random. For if it were not, that would imply something systematic about it. So the data could actually be represented or coded more efficiently, resulting in less transmitted bits, which would then have a random distribution.

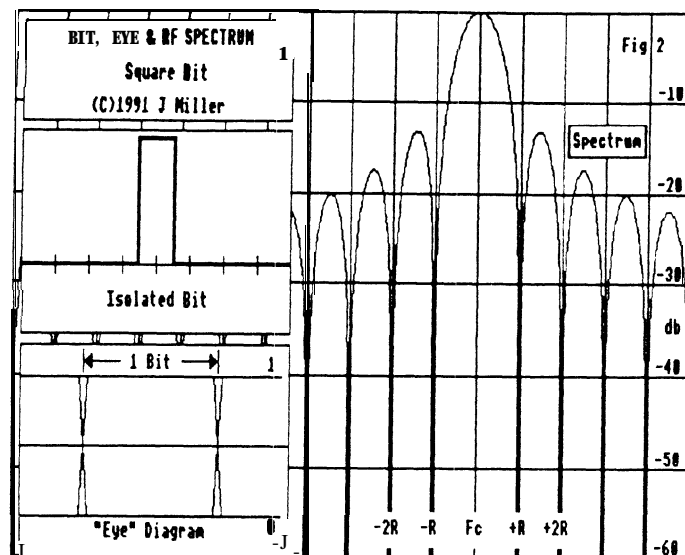


Fig 2. Rectangular bit, its eye and spectrum. See text for details.

UNDERSTANDING BIT, EYE & RF SPECTRUM PLOTS

Now let's examine some representative cases. What happens if we modulate the carrier directly with unprocessed rectangular bits?

Fig.2 shows three things a) isolated rectangular bit, b) its RF spectrum and c) what the data stream looks like on an oscilloscope. Of course the bit edges are in reality almost invisibly fast, so I have emphasised them slightly.

a) Isolated Bit. The horizontal axis is time, and is marked off in eight bit intervals. (Eight are shown). The vertical is voltage.

b) RF Spectrum. The horizontal axis is frequency, and is marked off in units of the bit rate R bps. For example, if the data rate is 1200 bit/sec ($R = 1200$), the vertical markers are at 1200 Hz spacing. The mid point of this spectrum is of course the RF carrier frequency. The vertical is marked off at 10 db intervals.

c) Oscilloscope diagram.

1. The horizontal axis is time, and spans a total of two bits; so it's expanded x4 compared with the isolated bit. The vertical is voltage. You are to imagine the 'scope is triggered by a local bit-rate clock. The resultant pile-up of bits is what you see displayed. It is the sum of many positive and negative isolated bits, each displaced in time by one bit.

2. If this display is the received waveform, it's what is sampled by the data detector to decide if a "1" or a "0" has been received. Usually this decision is taken at the mid-time point of the received bit by a sampler. Above the horizontal line is the regime of "1"s, below "0"s.

3. This display is also universally known as an "eye diagram". It is a concise representation of the quality of the received signal. The less confused the trace, the greater the distinction between high and low at the mid-bit sample point, then the more reliable (less error prone) bit detection will be.

4 These traces show no channel noise. That is to say, the signal to noise ratio is assumed high. If there were noise, the traces would be blurred. If the noise were momentarily large enough to cause the waveform accidentally to cross the middle horizontal line, an erroneous detection decision would be made, leading to a received bit error.

RECTANGULAR BIT

Returning to our rectangular bit shape. Two things are clear from figure 2; first, the extremely high fidelity of the data waveform and second, the price for this, profligate use of bandwidth. The diagram shows significant energy spreading well beyond 10 times the data rate. Indeed, as far away as 30 R (R = bit rate), the sidelobes are still only -40 db down. As sufferers from computer hash will know, rectangular bits are effective noise generators.

In some applications (for example spread spectrum communications) this might be the desired result. In the amateur radio environment where we want to get as much data through the limited bandwidth of our receivers, more finesse is required

Obviously we need to filter the data stream before transmission, because this will attenuate those sidelobes, and constrain the bandwidth.

So the key issue, indeed perhaps the central problem of all data transmission, is "what form should this filter take"?

PERFORMANCE OF A SIMPLE R-C NETWORK DATA FILTER

We tentatively try a simple filter. Let's pass the data stream through an R-C network with a -3db point equal to half the data rate. The result is shown in figure 3.

Look at the spectrum; while the main lobe is hardly changed, the sidelobes are reduced by 10 db or more, which is what we wanted. Look at the isolated bit; it has become rounded. But note particularly that its duration now exceeds one bit. In fact it's roughly two bits long.

This means that successive bits overlap. In turn this brings a new potential design problem; inter-symbol interference, or ISI. Now we find that not only do we want spectrum control, but we have to do it with the constraint that successive overlappir.. bits must somehow not interfere with each other.

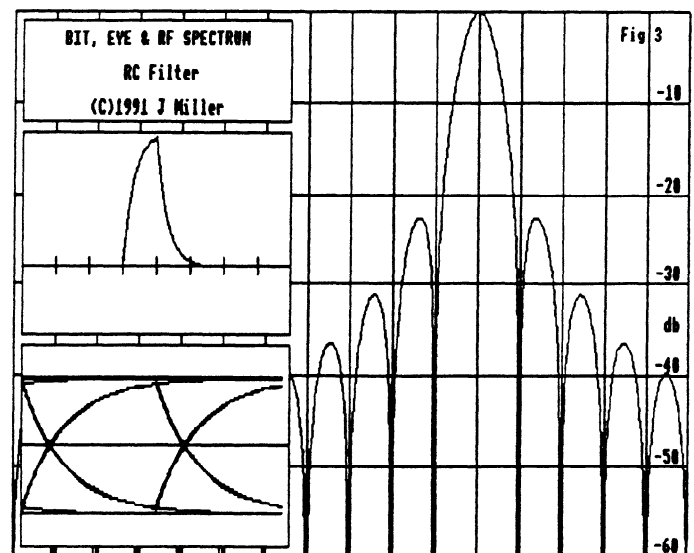


Fig 3. Rectangular bit through R-C filter has classic exponential rise and fall, and RF sidebands are reduced slightly.

Finally, look at the 'scope trace. The line at the top corresponds to the sequence 111..., that at the bottom 000... The sweeps from top to bottom are caused by ..10.. and ..01.. transitions and so on. You can also see that the traces are not confused; highs and lows of successive bits are separable, so there is no significant inter-bit interference.

INTER-SYMBOL INTERFERENCE REVEALED

In an attempt to attenuate the sidelobes more, let's now double the RC filters time constant. That is, the -3db frequency is now 1/4 the bit rate ($R/4$). See Figure 4.

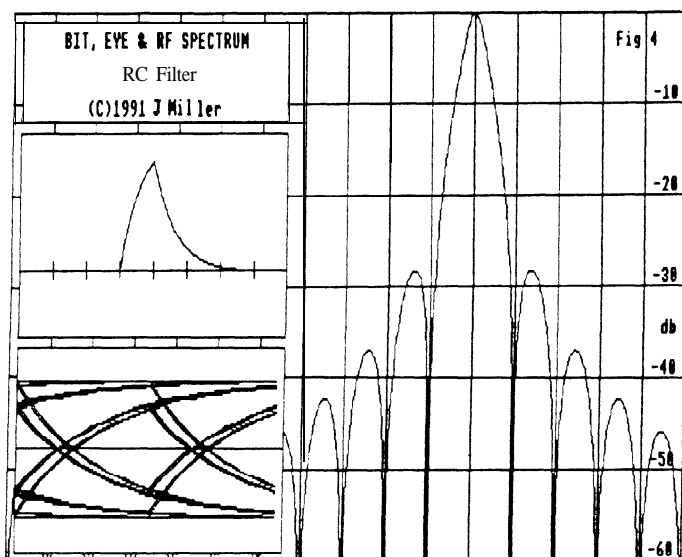


Fig 4. But too much ill-designed filtering gives inter symbol interference (I.S.I.).

Though the sidelobes are reduced, the main lobe is also rather pinched. But the real casualty can be seen in the 'scope trace. Under certain conditions (for example the sequence ..010.. or ...101...) the voltage barely gets half way up to the top line. Sure, bits are still distinct and detectable, but the noise margin is drastically reduced by some 50%.

Analytically, the reason for this is seen in the isolated bit diagram of figure 4. The bit starts off at $T=0$. At $T=1$ it reaches its peak. At $T=2$, it hasn't fallen back to zero again, but has a value approximately 1/3rd peak. This aberration is the sole cause of the inter-symbol

interference apparent in the 'scope trace. At $T = 3, 4$ etc, the voltage is back to zero again.

From this observation we can formulate the requirement for a bit shape that guarantees no ISI. There should be a peak of unity at $T=0$. Then at all other exact bit points ($T = -2, -1, +1, +2$, etc) the isolated bit waveform should be exactly zero. What happens in between doesn't matter.

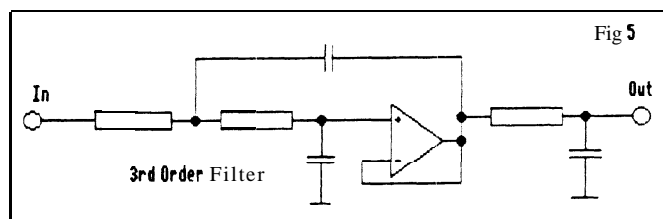


Fig 5. Packet satellite FO-20's 3rd Order Data filter is similar to this. Waveshapes as figure 6.

FO-20's DATA FILTER

An example of a bit shape that meets this requirement is that transmitted by the satellite Fuji-Oscar 20. The data filter is (I believe) a 3rd order Bessel type with a -3db point at 0.532 times the bit rate ($0.532 R$). "Bessel" filters driven by square waves have a nice steady rise, and negligible overshoot or ringing. In electrical circuit terms, the filter is merely 3 resistors, 3 capacitors and an op-amp. The filter is shown in figure 5.

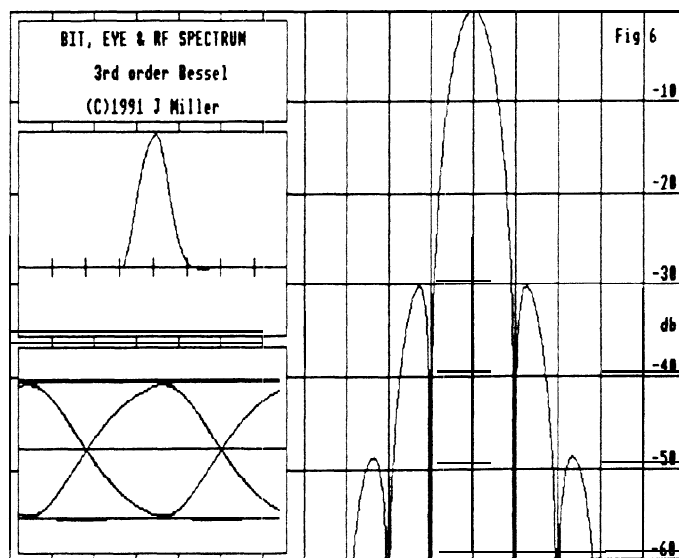


Fig 6. Characteristics of FO-20 Data

The spectrum is quite reasonably controlled; 99.9% of the energy is contained within a bandwidth of 1.75 R (as compared with 15 R of the rectangular pulse). Since the FO-20 bit rate is 1200 bps, this means that most of the signal occupies an effective RF bandwidth of the order of 2.1 kHz, so it stands a fair chance of passing through the SSB receiver filters without significant distortion.

The isolated bit shape is zero at all T mints except at one, and this leads to a nice 'scope trace. The origin of the term "eye diagram" can be seen here, as the trace is supposed to resemble a human eye. Notice that as a consequence of negligible inter-symbol overlap in the isolated bit, the 'scope trace has perfect bit convergence at the sample point.

The isolated bit has a duration of just 2 bits. This means that there can be only $2^2 = 4$ trajectories on the 'scope trace (00, 01, 10, and 11). The 01s and 10s show up as a very clean zero crossover. If the RF bandwidth were narrower then the isolated bit would span more than 2 bits, so the 'scope trace zero crossings would show dispersion.

AND NOW - "RAISED COSINE"

The term "raised cosine" means several things, a fact which causes a lot of confusion. A raised cosine is just that. A cosine function raised above its mid point. The formula for a raised cosine is

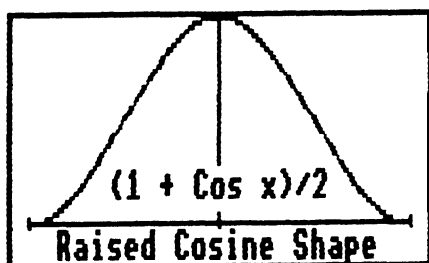


Fig 7a. A raised cosine shape. There's nothing mysterious about a raised cosine. Just get out your calculator and plot $y = (1 + \cos x)/2$ for $x = -180$ to 180 degrees!

$(1 + \cos x)/2$. Drawn as a graph, it's merely a shape. But it has a number of analytical properties that make it convenient to employ. These are simplicity, left-right and upper-lower symmetries.

A RAISED COSINE BIT

"Raised cosine" can for example, describe the shape of the isolated bit - see figure 7. Since the isolated bit is cosine shaped, the 'scope trace is composed entirely of sine waves and straight lines. Indeed, the bit sequence ..10101010.. actually creates a pure tone at a frequency R/2 at baseband, or at RF, two frequencies spaced by $\pm R/2$.

The isolated bit shape is very similar to FO-20's, occupying 2 bits. Not surprisingly the spectrum is very similar too. In fact 99.9% of the energy is contained within 1.69 R (compared with 1.75 R for FO-20). An important practical point to remember is that the shape of the bit is completely specified in time. This means that it can be precisely generated using real finite hardware. The microsats AO-16/WO-18/LO-19 use a filter not unlike this.

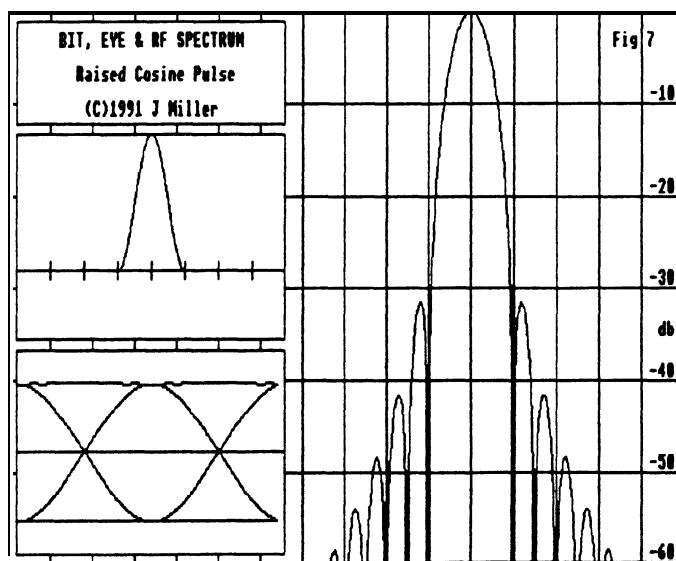


Fig 7. Raised cosine bit TIME shape used (in principle) on PacSats AO-16/WO-18/ LO-19.

THE RAISED COSINE SPECTRUM

Alternatively "raised cosine" can also be used to describe the shape of the spectrum. This is illustrated in figure 8. It doesn't look much like a raised cosine spectrum because of the logarithmic scale, but the response is unity (0db) at $f=0$, 0.5 (-6db) at $f=R/2$ and exactly zero at $f=R$, the bit rate. 99.9% of the energy is contained within $1.56 R$.

Because the spectrum is absolutely band limited to $\pm R$, the isolated bit shape that gives rise to that spectrum has infinite time duration. This means that it cannot be synthesised with real finite hardware. On the other hand, as figure 8 shows, it has negligible amplitude beyond $T = \pm 2$ bits, so that discarding the tails has little effect other than to bring in some very minor spectral sidelobes. So synthesis turns out to be highly practical.

You will also notice that the isolated bit also has precisely the desired properties for zero inter-symbol interference. Is this fortuitous? Certainly not! In fact it is a direct consequence of the two symmetries of the cosine shape mentioned earlier. That's why the raised cosine spectrum is widely used in data transmission system design.

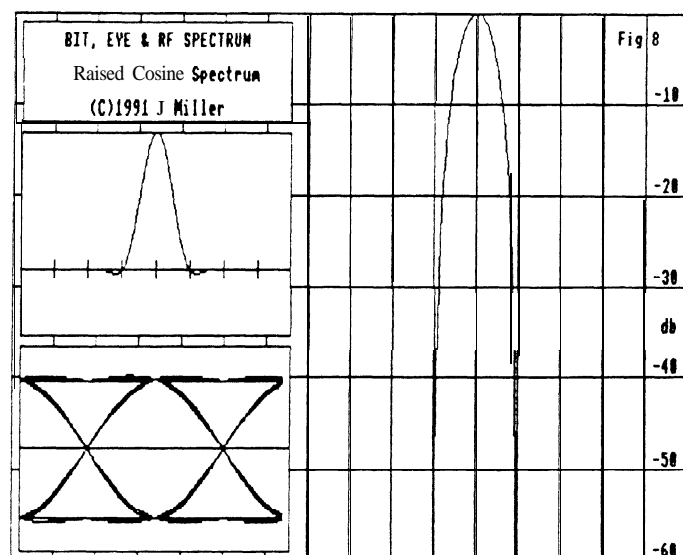


Fig 8. Raised Cosine SPECTRUM Shape. There is a family of these shapes, with the raised cosine part centred on the middle of each side of the spectrum.

9600 BAUD PACKET RADIO & UOSAT 14 -A NARROWBAND RAISED COSINE SPECTRUM

As a practical example of "raised cosine" spectrum shaping consider figure 9. Here only the middle 3/8ths of each half of the spectrum is given a raised cosine shape. From $f=0$ to $5/16 R$ the spectrum is flat (0db), from $5/16 R$ to $11/16 R$ it follows the cosine shape, and from $11/16 R$ onwards the spectrum is zero.

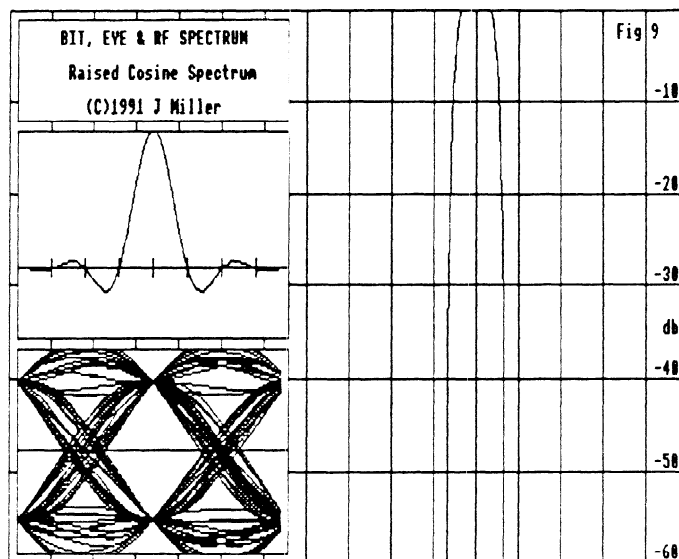


Fig 9. UOSAT 14 has almost "rectangular" raised cosine spectrum (see text). The narrow single main spectrum lobe implies a bit that is some 8 bit periods long. There is no I.S.I.

This is used in the baseband filtering of UOSAT-3/ Oscar-14 which uses the G3RUH 9600 baud Packet Radio modem [1]. The isolated bit shape spans some eight bit periods. That is to say a bit transmitted "now" will actually have an influence on up to eight other bits, four before and four after "now". Note also that the isolated bit has the desired form for minimising inter-symbol interference. A peak at $T=0$, zero at all other T points. So the received "eye" shows good convergence at the sample point.

The spectrum is almost rectangular ("brick-wall") and sidelobe free. 99.9% of the spectral energy is contained within $1.2 R$, or ± 5.6 kHz at 9600 bps.

The zero-crossing dispersion is a direct consequence of the narrow bandwidth, and great care is needed in the design of the receiver bit-clock recovery circuits that perform time bit detection. You can see that sampling only 1/8th bit away from the convergence point means bit detection half way down to the zero threshold, and so the instantaneous noise tolerance would be reduced by 6db. Narrow bandwidth bit-clock recovery circuits are essential. For implementation details see [1].

The actual signal transmitted by UO-14 is slightly different to that shown; it is pre-distorted (in time) so that when it passes through a real bandwidth limited receiver the final "eye" is as illustrated.

In the G3RUH/PacComm/Kantronics/Tasco Tereleader embodiment of this modem, transmit bit-shape is looked up from a table of values in an Eprom and passed to a DAC. This data filter is known as an F.I.R. for Finite Impulse Response. UO-14 uses a tapped digital delay line (shift register) and summing op-amp; Eproms do not travel well in spacecraft. This mechanisation of an F.I.R. is called a Transversal Filter.

In UO-14 the signal also EM modulates the carrier (as opposed to PSK). FM is simple to generate, much simpler to process on receive in the presence of mistuning such as doppler shift, and is very robust. But

FM does require slightly stronger signals, 6-8 db perhaps.

Finally note that this precision data shape could trivially generate precision BPSK via a DSB balanced modulator. Such PSK would have all the narrow-band and non-ISI attributes detailed above.

TRANSMITTER AND RECEIVER FILTERING IN SERIES

So far discussion has mostly been about spectra and bit shapes in association with transmitters. It was implicit that the receiver was infinitely wideband. For analytic purposes, if there were any RX filtering it could be assumed to be merged with the transmit filtering.

In many amateur applications this is indeed the case. For example, the current flock of PSK packet satellites confine their bit rates to that which can be passed through a typical SSB receiver. RTTY users assume the same thing.

We now ask the important question; if the overall filtering were to be deliberately split between transmitter and receiver, what should the form of the filters be?

We know from all the above what kind of overall response is required. But how should we partition it?

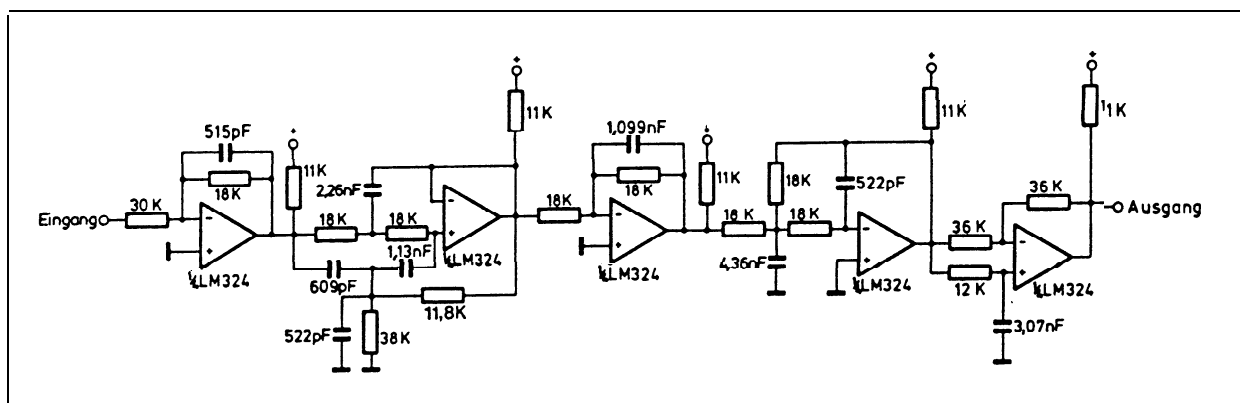


Fig 10. Data filter used in RUDAK-2 "RSM" transmitter. This comprises three cascaded sections; a 3rd order Cauer, a 3rd order Butterworth and an all-pass (refer to text). This transmit filter is used in association with an almost identical filter at the receiver. These filters are thus "matched". For spectrum and bit shapes see figure 11.

Obviously, the overall response can be apportioned in any ratio whatsoever. However one can prove that in the presence of random noise the optimum split, so as to maximise detector signal-to-noise ratio, is that with exactly half the frequency response in the transmitter, and the other half in the receiver. These are known as a "matched filter pair".

Conceptually, one can first imagine a desired overall frequency response dictated by inter-symbol interference requirements, spectral constraints and implementation ease. Then as it were, you take the "square root" of this, and put the resulting filter into both transmitter and receiver.

AO-21/RUDAK-2 "RSM" BIT SHAPING

A very good example of this is the bit shaping scheme used on one of the RUDAK-2 links of AO-21 [2]. Wittingly dubbed "RSM" Rectangular Spectrum Modulation by its creator, its transmit filter properties are shown in fig. 11 (left). Both transmitter and receiver have essentially identical

filters. This bitshape is used to phase modulate (BPSK) the RF carrier. Bit rate is $R = 9600$ bps.

99.9% of the spectral energy is contained within a bandwidth of $1.5 R$, which is consistent with the isolated bit span of about 3 bits. This means that there are about $2^3 = 8$ trajectories in the 'scope trace, which can be seen quite clearly. It is notable that this isolated bit shape is created using analogue filtering rather than more accurate and repeatable digital synthesis.

The transmit filter (fig. 10) has 3 sections, comprising a) a Cauer Chebyscheff type CC0315/41 which is flat up to a frequency $f = R/2$, and then plunges down to the zero at $f = R$. But above this there is a rise to only -15 db. So part b) is a 3rd order Butterworth that's smooth to the -3db point at $f = R/2$, and then -18db at $f = R$ and -36 db at $f = 2R$; c) an "all-pass" network that has a dead flat frequency response, but a tapered phase characteristic which is used to flatten out the filter delay vs. frequency curve. This makes the transmit bit shape become

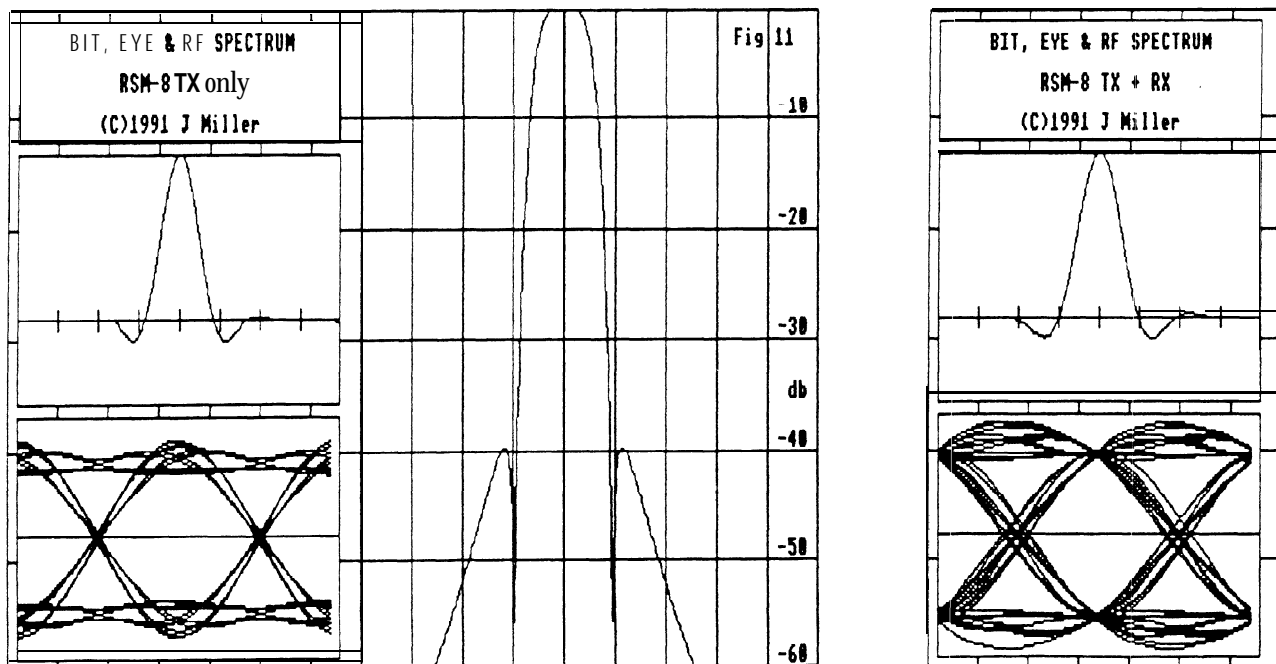


Fig 11. Performance of RUDAK-2's "RSM" TX filter (L) and signal after passing through complementary "matched" TX + RX filters (R). The eye is as wide as possible at the detection point, and regardless of neighbours, all bits have the same amplitude at this instant. (i.e. no I.S.I.)

left-right symmetric. This filter is not driven directly by the rectangular bit stream, but by short + or - pulses of polarity corresponding to the data bits' polarity.

When cascaded with an identical analogue filter in the receiver, the overall response is as per figure 11 (right). The overall isolated bit shape is almost perfect, spanning some 5 bits with remarkable symmetry. The 'scope trace shows an excellent eye. The sample point convergence has about +/- 8% scatter due to the slight non-zero values of the isolated bit shape at $T = -2$ and $T = +2$.

THE FUTURE?

The 9600 baud RSM-PSK implementation is near optimal in performance, probably within 1 db of the theoretical limit. [approx. $20 \log(1 - 8/100)$]

It is flying on RUDAK-2/AO-21 so AMSAT experimenters (i.e. YOU) can evaluate the effectiveness of high speed PSK with or without coding [3] from the associated on-board Harris RTX2000 RISC processor. This will help determine what is practical and realistic for amateur spacecraft and other digital links of the future.

The G3RUH 9600 baud FM system is not only in daily use by many satellite stations on UO-14, but also hundreds of terrestrial packet radio links world wide. It represents a design that pushes the fastest possible binary data through a basic FM radio. 9600 bps throughput is astonishing to behold, especially on a full duplex satellite link.

Who will predict that one day even our voice transponder links will be entirely digital. Amateur ISDN? Now there's a challenge!

REFERENCES

1. Miller J R, "9600 baud Packet Radi Modem Design", Proc. 7th ARRL Computer Networking Conference, 1988 Oct, pps 135-140
2. Meinzer K & Haas W, "RUDAK 2 - The Radio Links", Amsat-DL Journal No. 1/17, March 1990 pps 9-12. (German). English translation in Oscar News No. 83, 1990 June, pps 16-21
- 3 Miller J R, "Shannon, Coding and the Radio Amateur", Oscar News No. 81, 1990 Feb, pps 11-15

CONTACTS

AMSAT-UK, London E12 5EQ, England. (Oscar News).

AMSAT-DL, Holderstrauch 10, W-3550 Marburg 1, Germany. (A-DL Journal)

PacComm Inc, 3652 W Cypress St, Tampa, FL 33607-4916, USA. (NB-96 Modem)

Kantronics Co Inc, 1202 E 23rd St, Lawrence, KS 66046, USA. (DE9600 modem).

Tasco Tereleader, 3 8 Minami-Youchi, Higasibata, Anjo-City, Aichi 444-12, Japan. (TMB-965 modem).

The Ottawa Packet Interface (PI):
A Synchronous Serial PC Interface for Medium Speed **Packet** Radio

Dave Perry, VE3IFB

Packet Working Group
Ottawa Amateur Radio Club
P.O. Box 8873
Ottawa, Ontario
K1G 3J2

ABSTRACT

This paper describes the design and implementation of a synchronous serial interface card using DMA for IBM PC compatible computers for use in amateur packet radio applications.

1. Background

When I became involved with the local packet radio group a couple of years ago, state of the art was to use the Heatherington 56K modem [1] with a modified terminal node controller (TNC) for the PC interface. This arrangement was far from ideal. While the radio channel ran at 56 kbit/s, the TNC to computer link ran at 9.6 kbit/s. Phil Karn had written a special driver for the DRSI PCPA card to run at 56 kbit/s. [2] However, because the hardware required programmed I/O, the driver would lock up the PC each time a carrier was detected on the radio channel. Clearly what was needed was a new piece of hardware to perform the interface function. The requirements were identified:

- A low cost synchronous interface for the IBM PC and compatibles (even the 4.77 MHz variety).
- Targeted at the end user in a network (as opposed to linking or packet switch applications)
- Easy connection to the Heatherington modem
- Software support for the NOS TCP/IP package [2]

2. Design Alternatives

Two approaches were considered to produce an interface with improved throughput and to remove much of the processing burden from the host computer. The first was to use a co-processor to handle the synchronous data, with a dual ported RAM arrangement to communicate to the host machine. This would have been a scaled down version of the "Awesome I/O" interface proposed by Chepponis and Mans [3]. This was rejected as being too capable (and therefore costly) for what I was after - an inexpensive interface for the end user.

The selected method was to use direct memory access (DMA) to transfer the packets to and from the PC memory. This method drastically reduces the processor load. The host machine can set up a packet for transmission, start the DMA transfer, and go away to do other things while the transfer takes place. A similar arrangement exists for receive. The host processor is interrupted only after a packet has been transferred into memory. Of course, you never get something for nothing. The price paid in this case is that each DMA transfer steals a few bus cycles which could otherwise have been used by the host processor, but this is a very acceptable trade-off.

3. Half Duplex vs Full Duplex DMA

This interface was targeted at the end user in a local area network (LAN) configuration. There was therefore no requirement for the interface to be able to transmit and receive data at the same time, since there can be only one user of the LAN at any instant, DMA channels are a very limited resource in the PC architecture. There are only 3, and 1 or 2 of those are often used by other interface cards such as disk controllers. For these reasons, the card was designed to use half duplex DMA, which requires only one DMA channel. DMA is used for both transmit and receive, and the direction of transfer is reversed with each exchange.

4. Circuit Description

The PI card uses a 28530 dual port Serial Communications Controller (**SCC**). **This** device has been described as the Swiss army knife of serial chips, It does, however, require a bit of glue logic to interface it to the PC bus. **U1** is a bidirectional bus buffer used to meet the loading and drive requirements of the bus. **U4** and **U11-F** provide buffering of additional bus signals. **U9** and **US-A** and **D** are used to delay the leading edge of the buffered **I/O** write signal for the SCC. This is required to meet the timing requirements of the NMOS version of the chip. **U3**, a **74LS138**, and the **16L8** PAL, **U2**, provide all address and chip select decoding.

U6-A and **US-D** provide a means of masking DMA requests independent of the internal registers of the SCC. This was needed because DMA and processor accesses to the SCC occur asynchronously, and it is possible to violate the minimum time between accesses required by the chip (6 times the clock period plus 200 ns). By disabling **DMA** accesses for a short time before and after a processor access, this constraint can be met.

The 8253 timer chip, **U8**, provides an independent timer for each serial channel. One timer in this chip is used as the prescaler for the remaining two. The timer outputs are connected to the **CTS** pin of each serial channel. In this way, the CTS interrupts can be used to time events.

U10 provides a divide by 32 function for baud rate generation for the B channel. This was included so that it will be possible to write a full duplex driver for the B channel. There is no divider for the A channel, because it is primarily intended to be externally clocked (by the Heatherington modem, for example), and because the DMA support for this channel is half duplex.

There is a transmit watchdog circuit for the A channel, consisting mainly of **U11-B**, **C2**, and **R4**. If **RTS** is asserted for more than approximately 10 seconds, the circuit will disable transmit, **Q1** gives an open collector output for keying a transverter. **Q2** can also be used to give a separate open collector output for the **RTS** line. This is intended to drive the **Heatherington** modem, which has a pull-up resistor on this line. This will prevent the transverter from being keyed up when the host computer has been turned off. **U11-A**, **B** and **C** are the clock oscillator and buffer for driving the SCC at 3.68 **MHz**. The timer chip is clocked at half this frequency by **U6-B**. Position **U12** can either be jumpered across to provide a **TTL** level interface to the B channel, or it can be stuffed with a Motorola MC145406 RS232 transceiver chip.

5. Software

The PI card driver has now been integrated into NOS. The source code for it (as with the rest of NOS) is freely available.

6. Conclusions and Future Directions

File transfer rates of up to 5000 bytes/sec have been achieved under controlled conditions, while rates in the range of 3000 to 4000 bytes/sec are representative of real world performance on our network. While this is a great improvement over the rates of a few hundred bytes/sec which were previously attained, there is still room for improvement. The transmit delay time required by the modems is on the order of 15 ms and should be reduced, since it has now become the biggest factor limiting performance.

The interface has also been successfully tested at data rates up to 250 kbit/s over a wired connection with a special version of the driver.

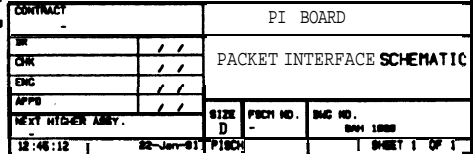
As a result of volunteer efforts by members of the Packet Working Group of the Ottawa Amateur Radio Club, a number of PI cards are now in use around the world. Contact the club for information.

On the software side, there have been requests for a packet driver compliant version of the driver, and for G8BPQ front end support. I welcome correspondence with anyone who wishes to tackle these challenges.

7. References

1. Heatherington, Dale A., WA4DSY, "A 56 Kilobaud RF Modem", *ARRL Amateur Radio 6th Computer Networking Conference*, pp. 68-75, Redondo Beach, California, August 29, 1987
2. Karn, Phil, KA9Q, "Amateur TCP/IP in 1989", *ARRL Amateur Radio 8th Computer Networking Conference*, pp. 114-118, Colorado Springs, Colorado, October 7, 1989
3. Chepponis, Mike, K3MC and Mans, Bernie, AA4CG, "A Totally Awesome High-Speed Packet Radio I/O Interface for the IBM PC XT/AT/386 and Macintosh II Computers", *ARRL Amateur Radio 7th Computer Networking Conference*, pp. 36-40, Columbia, Maryland, October 1, 1988

124



CLOVER-II: A TECHNICAL OVERVIEW

Raymond C. Petit, W7GHM
P.O. Box 51
Oak Harbor, Wa. 98277

ABSTRACT

The CLOVER-II four tone-pulse signal, methods of synchronization, control and data modes, Reed-Solomon coder, and transmission protocols are described.

INTRODUCTION

In a paper at last year's conference¹ I reported results of some on-the-air testing of a new HF data communication system I have been developing over the last few years. These tests verified my expectation that the "Cloverleaf" system would provide very worthwhile performance improvements over **any mode** in present amateur use. The original version has been named "CLOVER-I". It is distinguished by having data pulses at one center frequency, practical channel spacing of 100 Hz, error-corrected throughput of well over 100 bits/second in the best of conditions, and a requirement for very high frequency precision in the radios.

In August of last year I teamed up with Bill Henry (K9GWT) of HAL Communications. We decided to pursue development of a second version which we named "CLOVER-II." This version has 500 Hz channel spacing and error-corrected throughput of over 500 bits/second in the best conditions. It will work with existing synthesized radios of recent design. First production units are scheduled for delivery by early 1992. A patent application covering both versions was filed on July 8 of this year.

THE CLOVER-II "CARRIER"

The "Carrier" of the CLOVER-II signal is a sequence of four overlapping pulses having smoothly-shaped amplitude envelopes and ascending center frequencies. Each pulse has a duration of 32 ms and successive pulses are offset by 8 ms. The pulse envelopes are Dolph-Chebyshev functions. The power spectrum of each pulse is at least 50 dB down outside its subchannel width of 125 Hz. The pulse at the lowest frequency begins at the frame origin. The pulse 125 Hz higher begins 8 ms after the origin, the pulse 250 Hz higher begins 16 ms after the origin, and the pulse 375 Hz higher begins 24 ms after the origin. The peak-to-average power ratio (crest factor) of this carrier is only 6 dB.

¹Petit, R.C. (W7GHM), "The 'Cloverleaf' Performance-oriented HF Data Communication System", ARRL-CRRL 9th Computer Networking Conference, page 191

The CLOVER-II carrier has been designed for 500 Hz channel spacings. The spectrum of the CLOVER-II signal is the **same**, data or no data, and regardless of the data speed. Four CLOVER-II signals can fit into the **2-kHz** space now required for the same level of channel isolation with the narrow-shift FSK modes. **Fig. 2** shows measured power spectrum data for CLOVER-II, HF packet, and **AMTOR**. As a test I set up one of the prototype CLOVER-II controllers to report the total power level of an incoming CLOVER-II signal in the lab. Then I moved the transmitter signal up exactly 500 Hz and got a second measurement of the total power in the original receive channel. There was a 52 **dB** difference.

One may logically ask, "**Why** not just stick to one tone pulse frequency, as you did in CLOVER-I, and speed it up by a factor of 5?" CLOVER takes a hint from CW. The duration of a pulse is much **longer** than the uncertainties in time-of-arrivals due to **multi-path**. The energy in one CLOVER pulse is concentrated in a very narrow band. When selective fade sets in, all of the pulse more or less fades in unison. The very high subchannel isolation **permits** the receiver to decode each subchannel independently, recovering data even if a subchannel amplitude is many **dB** below that of its neighbors. Compensation for channel variability (adaptive equalization) is very easy with this arrangement.

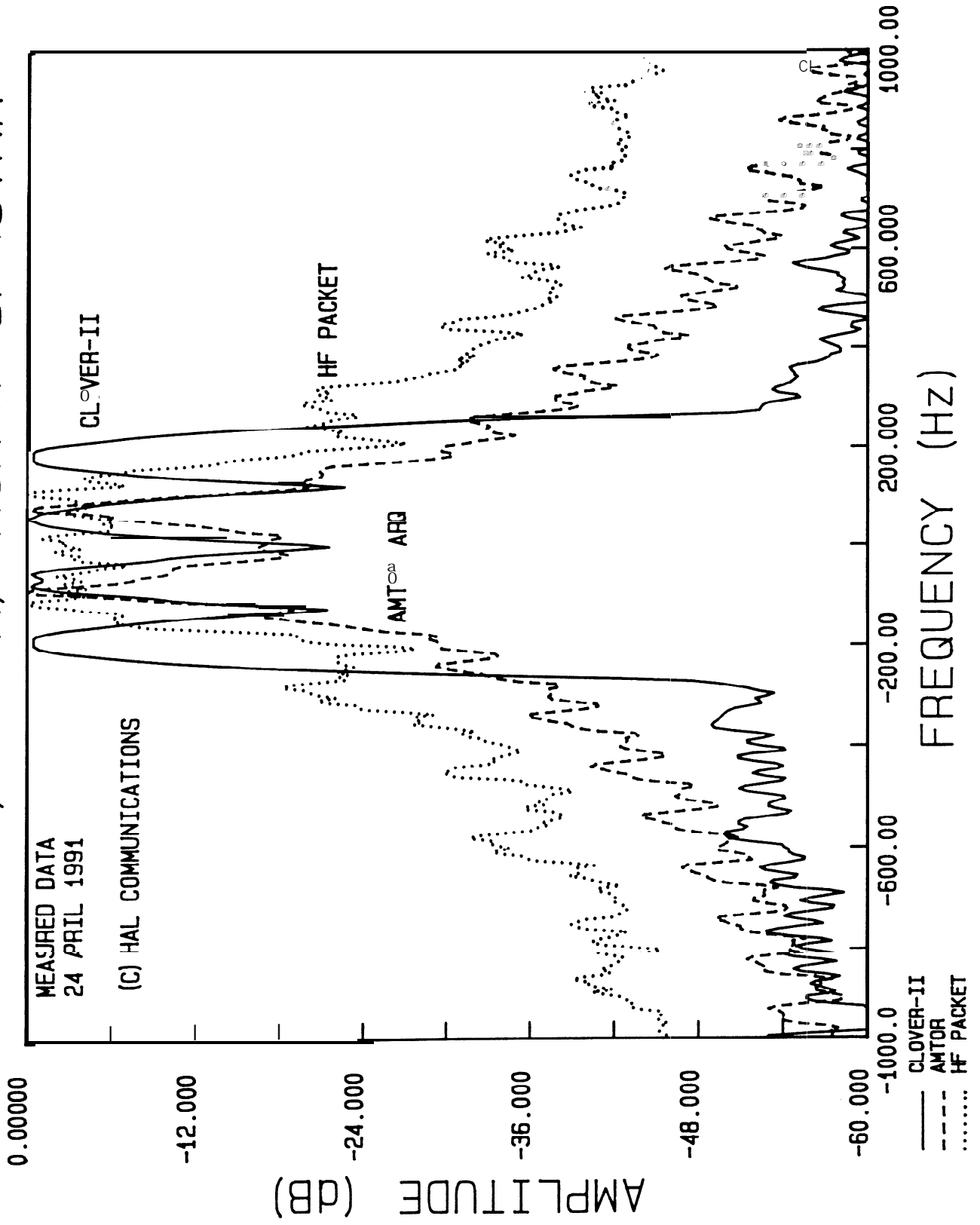
TIME AND FREQUENCY SYNCHRONIZATION

CLOVER is a synchronous data-transmission scheme. Two levels of synchronization are required. "Frame sync" positions the receive data-sampling processor to the instant of maximum S/N of the four pulses in a frame. "Epoch sync" establishes the boundaries of Reed-Solomon codeblocks. Both forms of sync are obtained during the link-establishment process and are valid until a link reset occurs, either for station ID every 10 minutes or to recover from a long series of decode failures.

Frame sync is obtained from a **1/2-second** observation of the CLOVER-II carrier with a signal averaging process applied to the pulse magnitudes. Epoch sync plus "target" station identification is obtained from a crosscorrelation between a byte pattern in which the target station **callsign** is embedded, and the "**MYCALL**" data of each receiving station.

CLOVER uses differences in phase between successive pulses in each subchannel as its primary data-transfer mechanism. At the higher speeds, one of 16 phase levels must be distinguished every **32 ms**. A frequency offset causing a phase drift of more than plus or minus **1/32** of a cycle in **32 ms** will produce errors. This is only 1 Hz offset! By observing the phase changes near the maximums of each of the four pulses in a frame, the CLOVER-II receiving processor determines the frequency offset between the two stations and corrects for it. A tuning indicator will show this offset, and an alarm will be sent to the application program if this offset exceeds 10 Hz. Later versions of the CLOVER-II software may make provision for automatic fine tuning of the receiver. (However, a

CLOVER/AMTOR/PACKET SPECTRA



far better alternative, in my opinion, is to make a stable synthesized radio which can be phaselocked to an accurate standard. **I provided diagrams** for a simple single-frequency stabilized radio last year.² Links using stabilized radios at both ends will deliver better performance.)

DATA AND CONTROL MODES

CLOVER-II has a suite of 7 modulation modes. These range from slow and very robust modes for the very poor propagation conditions up through the fast and fragile modes for the very best band conditions.

The first mode is actually a set of three. In these modes, phase information is not used: only the subchannel amplitudes are used. Data or control information is sent by omitting two of the four pulses in a frame. In the data-sending variant, it is a **dual-diversity** format, 1 bit/frame, 31.25 bits per second. This mode is useful when conditions are so poor that phase-encoded information fails altogether. The other two variants are used for link maintenance control signals.

The slowest phase mode is "Quad **Inband** Diversity Binary Phase." Each of the four pulses in a frame carries the ~~same~~ data bit. A **smart** diversity combiner in the receiver chooses the channel having the combination of highest amplitudes and most nearly ideal phase differences and reports its data as the best estimate. The raw data speed (not error-corrected) is one bit per frame, 31.25 bits per second. This mode has the ability to estimate the channel BER.

The next four modes encode data in phase differences only, using all four pulses. They use 2, 4, 8, and 16 phase levels, respectively, for raw data rates of 125, 250, 375, and 500 bits per second.

The fastest mode adds 4 amplitude levels to the 16 **phase-level** mode to get raw speeds of 750 bits per second. The incoming signal will have to be very strong and the path very stable. The receiver AGC will have to be off. I don't expect this mode to be usable very often.

REED-SOLOMON CODING AND BLOCK FORMAT

CLOVER uses large block-size Reed-Solomon coding without interleaving. All processing is done on **GF(2E8)** using fast transform methods. The block sizes are 17, 85, and 255 bytes, with symbol-error correcting capacities of 1 to as many as **1/4** the total number of symbols in the block. The number of data bytes in a block is (approximately) the block length minus twice the maximum number of errors it can correct. Two bytes of every block are reserved

²Petit, R.C. (**W7GHM**), "Frequency-Stable Narrowband Transceiver", ARRL-CRRL 9th Computer Networking Conference, page 195

for link maintenance data. Depending on the modulation mode and block size chosen, transmission of a single block will take from about one second to about one minute.

LINK-LEVEL PROTOCOLS

Three link-level protocols are in development. The first is a point-to-multipoint somewhat like AMTOR FEC. The second is the 'dialog' protocol resembling AMTOR ARQ. The file-transfer protocol is similar to the dialog protocol except for two features. User data is sent in only one direction, and single transmissions may contain many codeblocks.

MONITORING

If a CLOVER-II controller receives a call not directed to it, or hears a CLOVER-II signal already on the channel, the controller automatically goes into monitor mode, although if the user or application program does not want to receive monitored text, the text output can be suppressed. The controller will not transmit while it is in monitor mode, thus keeping the channel clear for the session already in progress. When a session ends, both controllers on the closing session transmit a distinctive 1-second control signal announcing their relinquishment of the channel. The receipt of this signal, or a sustained "no carrier detect" condition,, enables transmit states.

IMPLEMENTATION

The production version of the CLOVER-II controller will be an IBM-AT compatible expansion card. Supplied with it will be a terminal program similar to HAL's "PC-AMTOR" for keyboarding users. A "TNC EMULATOR" port on the rear panel will permit packet BBS sysops to convert their HF ports to CLOVER-II by simply moving one RS-232 cable.

Improving the Packet Mail Transfer System

Immediately Realizable Goals and Posturing for the Future

Brian B. Riley, **KA2BQE**
Radio Amateur's Telecommunications **Society**

The current amateur packet radio message transfer system runs extraordinarily well in view of a myriad of conflicts, mistakes, equipment failures, et al. that plague it every hour of every day. The volume of traffic is rising almost exponentially. It could reach critical mass and 'break' the system before technological solutions can be put into place. There are, however a number of things we can do within the constraints of current software and facilities to relieve much of the pressure and streamline the system for the future. Beyond that, there are several things that the codesmiths of the packet BBS world could implement reasonably quickly to further alleviate the problems.

Forwarding Headers

The current accepted **NK6K** header and its **WA7MBL/KA2BQE** variant should be replaced by a simpler format containing just the bare essentials. This format, shown below, has already been adopted by significant number of systems across the USA and several of the prominent international mail forwarding stations.

R:910710/1234z 12345@KA2BQE.#NWVT.VT.USA

No QTH, no Postal Locator, No advertisement for your BBS or software writer, nothing but time/date stamp, message number on that system and the full hierarchical address of the system. (The missing continent indicator is not a typo, but is a matter to be addressed in this paper.)

The justification for the removals/modifications is:

- a. The major consideration for the header is the 'accountability' for the handling of the message demanded of us by the FCC and its analogous bodies in other countries; to wit, that the message clearly show each station that handled it and be identifiable in each station's record keeping. With the hierarchical addressing universally being used and included plus the local message number these requirements as well as our own routing analysis is fully served.
- b. It would be most efficient to simply state the time and assume 'Zulu' time and annotate if it is other than that. However, there is no strong standard on a worldwide basis so for the moment we should all try to set "Zulu" time and annotate with the **"z"** to so indicate it.
- c. The indicator for **'@:'** becomes **'@'** and the **'#:'** is no longer needed.
- d. The WORLI Packet Bulletin Board System introduced to us the concept of White Pages (WP), which, in short, snoops through every message passing through a system. It takes note of the originating individual and the station from which the message was entered. It also takes note of every change in a local user's status in terms of QTH, zip-code, home BBS, name. This WP operation then formats messages once per day making note of changes and sends them, generally, to a regional WP server, who then summarizes all changes he received from the region and send them **onpv** to the national server as well

as back to the local stations. This information not only categorizes and records individual users but quite obviously the **BBSes** themselves.

For purposes of this paper I saved about **3800** messages that passed through my system over a few months. I ran a utility program that **I wrote** to convert headers to the format shown above against the message base. The least shrinkage to any bulletin was **36** bytes (a locally originated bulletin quite obviously); the greatest shrinkage was 1898 bytes (56 headers, originating in Europe and making several 'laps' around the continent before coming to Canada and to NWVT.) The average shrinkage was 801 bytes. More **important** was the average shrinkage as a percent of the total message size which was 29%; that is the messages shrunk to an average of 71% of their former size.

While it would seem that by removing this information from the headers we are removing **WPs** source of information we must remember that this information also flows in from each system based on USER Data files entries as well.

Querying the WP system will get any information that is **needed** on any BBS. The updates that get sent out based upon changes to the user data bases will be sufficient to keep the local/regional/national WP Server stations apprised of QTH and ZIP of the various BBSes.

Note for the future: we might wish to add some info fields to the WP database for **BBSes** to permit some additional information like, adjacent network node:, bbs code **type** and version. The WP system is ripe for enhancement to allow considerable information to be stored at the regional level for its local users. The national level WP database might store general pointer information to the regional WP database where more detailed information would be recorded/retrieved.

Several well intentioned kluges have been enacted to send the BID and message originator in the "R:" header. This is extraordinarily redundant. If this **information** were stored in a RFC-822 like header once in the beginning of the message it would save dozens of bytes per message. The use of RFC-822 headers is discussed further down in the paper.

Hierarchical Addressing and Forwarding

The current hierarchical addressing and forwarding was implemented on the basis of a paper written and presented at this conference several years ago. It **has** provided us with a framework that has allowed the forwarding system to grow in leaps and bounds without the need for every system to know every other. Indeed, today most **sysops** don't know anything about systems located in 80% of the states located **more** than **one** states removed from them; Further, they don't need to know it either.

There are however, a number of problems inherent with the original system proposed and implemented which were pointed out even then by a few, and now are recognized by most. The original format being used is:

host.local-org.locale.country.continent

The problem occurs in that parsing for purposes of routing occurs from left to right, or most specific to least specific. Most of the current systems go through lists seeing if they know anything about a given system, then only failing that does it move right to look at

local organization, state, then **country and** continent. The problems that occur are **manyfold**:

- There is a region in France that had the identifier "MA". Messages from the US that were supposed to head there were routinely being routed to **MAssachussetts** because the **"MA"** was evaluated before the **"FRA"**
- We had to get the kluge '#' prefixing numbered districts like Japanese prefectures and ' ' prefixing a British zones in order to keep American zip-codes routing from being applied. Once again because the numbers were evaluated before the country code.
- A ham like WINPR winters in Florida, summers on Cape Cod, and runs a BBS in both places. He goes through several weeks of misrouted mail each changeover. This is because all the systems along his paths "know" that he is one place or another and mail gets halfway through its proper route before it hits a system that didn't get the word and the message gets turned around. Again, this is because the **"WINPR"** gets evaluated before the "MA" or "FL."

The solution is obvious; the BBS code smiths get to work and get it all converted over to parse from least specific to most specific. To my knowledge PRMBS is the only system coded that way. But, even should all the other systems be recoded by Thanksgiving, the lead time on getting full world wide distribution to all systems will be horrendous.

There is however an interim solution which not only almost solves the problem but it makes the setting up of forwarding tables far easier for the sysops and most probably will speed up his forward process.

- a. Most important, remove all individual user and BBS CALLS from the various places in your forwarding route files for any regions beyond your own. Don't give the BBS a chance to send WINPR the wrong way, make it look beyond to the state first.
- b. Remove all the continent designators. The original seven proposed continent designators are not satisfactory. There are numerous problems in Latin Ameirca, Oceania and Asia posed by these. There was a proposal by Tom Clark, **W3IWI**, to either dump the continent designator all together or go to a four character designator and enhance the continent set. It is unfortunate that his proposal has met with only modest success and that success has unfortunately caused more confusion than it has helped anything. In the original proposal the country list was essentially an incomplete **ISO-3166** ALPHA-3 (3 letter alphabetic designator set). It contained everything we need now. There are but slightly over 200 countries on the list which is far less entries than most **PBBses** currently keep with all of the individual calls that are recorded.
- c. When the codewriters finally correct all of the code to parse properly from least significant to most significant element of the address then we can switch to the **ISO-3166** ALPHA-2 list and drop yet another character and also have addressing consistent with The Internet.

Message Headers

The time has come to recognize that we need to carry a certain amount of information in the text body of the message. In some of the above you can see several places where we carry the same information over and over **in** the **"R:"** headers that could be stated

once in the main body and be done with it. Since 1986 the PRMBS BBS code (**thanks** to the farsightedness of my colleague Dave Truli, **NN2Z**) has **utilized** a major sub-set of RFC-987 (RFC-987 is a common subset between RFC-822 and X.400) headers at the head of the text body of the message. While some of it was more show than useful, later a further subset was defined, its utility **has** time and again proven out because it provides for a place to pass information for which there was no provision in the current RLI defined forwarding protocol. Since RFC-987 parsing rules were fully implemented, additional fields could be added by any system and passed transparently through being acted upon only by those system that knew what to do with them. This highlights the chief flaw in the RLI protocol, that all systems must change to pass information on the SEND line and not lose it.

The minimum mandatory sub-set should be:

Date:
To:
From:
Message-ID:

In addition these should be handled, when present:

Expires:
Reply-To:
Priority:

Parsing headers should be done within the Internet fashion; **that** is to say, any line prior to an empty blank line which has a text string terminated by a colon ':', and a "space" is considered a 'header' and that the string will be examined, its objects parsed and acted upon if recognized, and passed through, untouched, if not.

There currently is a very real problem with worldwide distribution of messages of time value. They often wind up being delivered 10 days after their usefulness has expired. If we alter the manner of message creation for the user such that when he enters the message he is prompted for title, then 'Expiration?' and give him **the** option of entering **a number** (14 - fourteen days) or a date (1 1/14/91) or just hit enter (no expiration) In this fashion at least the users who regularly originate dated material will have it available and when their bulletins hit out towards the fringes and their time is up, they will die without requiring human intervention. The time it saves the sysops is time that can be used to take the trouble to examine manually other bulletins to see if they are worth keeping around, instead of resorting to **blanket** '4 days **and** gone' type bulletin management.

More on Addressing and Forwarding

There is a need to modify our ability to address mail. A user on a system in **NJ** should be able to send a personal message to **KA2BQE@VT.USA**. The systems should be able to route that message to the WP server for VT (most systems will now do this). What is needed is code at the server to recognize that a particular piece of mail needs re-addressing. In past, if there was anything in the @ field we left it alone. The codesmiths need to now discern between a general geographic location and a specific address (**ka2bqe@vt.usa** versus **ka2bqe@w1 koo.vt.usa**) so the bbs system knows whether or not it is permissible to re-address the message.

If the above suggestions about paring down forwarding files are followed and if all systems come on-line with a reliable **WP** server, it can work right now. As **part of a test**, **KB7UV**, in the New York City area, addressed a **message to "KA2BQE@VT.USA"**, it arrived in Northwestern Vermont in a timely fashion. The **"VT.USA"** was passed into **WA2SPL** in **Alburg**, VT, the regional mail server. and, **most important**, the regional WP server and it found its way correctly to **W1KOO** and then to **KA2BQE**.

Where the codesmiths can help out would be to modify bulletin distribution so that, say, a user in England planning to visit the state of New Hampshire in the **USA** could send a bulletin **SB INFO @ NH.USA** and request local repeater information from hams in New Hampshire. In other words the message gets treated like a single message to be routed only until it enters a BBS system that identifies itself with **NH.USA** and then it gets treated as a flood bulletin.

Logically this solution needs to be extended to multiple addressees for bulletins. That same user in England may be planning to visit Maine, New Hampshire, **Vermont**, and Northern New York. It is absurd to send four bulletins with four different **BIDs**. Likewise if the same **BID** is assigned to each bulletin the odds are just about 100 percent that only one bulletin would get through. Again, the RFC-822 style headers could be used:

To: info@nh.usa, info@vt.usa, info@me.usa, info@nny.ny.usa

Yet another need that would be nice to be able to handle would be the ability to include an individual user in the "To:" list and have a copy 'spun off' to him when the bulletin arrives in the area adjacent to that users address.

Improved SID Exchange

The Smart System ID (variously know as the **SID**, 'brackets message', or [...] exchange could be altered to produce a more efficient exchange. Currently, upon receipt of the **CONNECT** acknowledgement we await a BBS prompt. If during that wait we receive a **SID**, we continue to wait for the prompt and then return send our **SID** and await a BBS prompt, again. This double exchange can take **up** quite a bit of time. While it does not use up network bandwidth, it does chew up connect time to the BBS, at least 20 seconds and as much as several minutes. As an alternative, each system could **keep** a record of the other system's capabilities, as conveyed by the **SID**, and simply start sending as soon as it gets the "CONNECT" acknowledgement. **SID** would only be exchanged when a given station's records indicates that it has no knowledge of the connecting other station.

The principles of the new exchange are as follows:

- a. upon receipt of a **SID** waiting for BBS prompt or command any **station will return** their own **SID** to the other side.
- b. every **station starts out with clean record of what its connecting BBSes can do**. As such, when each is connected to or receives a **connect** from the other, **it is treated as a** conventional system and waits for/issues a **SID**. Once the exchange **takes place the** record for that system will reflect current capabilities and dictate future actions.
- c. When the system connecting out has had **a change in its capabilities, such that it must** issue a new **SID**, it simply will do so as its first command and receive a **SID** in return.

This process automates registration of capabilities with corresponding systems eliminating constant re-negotiation every session.

d. When a system connecting out has lost its records or for some reason is starting over, it will connect to a system that may 'remember' it as being capable and will not send a SID. In this case when the station connecting station reaches **timeout**, it sends its SID and waits one more time for response. Again a waste of time but only happens in this startup/restart circumstance.

e. When the connecting to system has lost its record, the **SID** will show up in **place** of the response to the first send command or to an **F>** reverse poll. The station should disconnect and update that system's record. Again a waste of time but only happens in this startup/restart circumstance.

Improved Message Transmission/Bulletin Negotiation

There are several things that need to be done here;

a. We currently send an Sx, where x is the message type. The types accepted are P for private (not that there is privacy but it more or less indicates person to person, or person to serve messages not really of interest to users other than the addressee), B for general interest bulletins, and T for traffic (NTS, MARS, **RACES,ARES**). Some systems also support an untyped message; that is a message from one user to another but is visible and readable by others.

Given ham radio's stated dedication to public service, one thing **we** have been remiss in providing is a method to move messages based upon some priority. We should extend the Send command to three characters Sxp where 'p' is the message priority (Normal, Immediate, Urgent).

For those systems with untyped messaging the underscore would **replace** the blank so an urgent untyped message would be S U. No typing would be **synonymous with** N for normal. Thus SPN and SP would be identical.

The codewriters will need to provide the software systems with a series of options that will have the BBS behave in some sysop **setable** fashion when it detects the presence of non-routine traffic.

This will be open to abuse. It should be handled by the sysop as a hold function of some sort if he finds a user regularly abusing the function. There is **no** other way to manage this as anyone can download and run PBBS code and do a simple **"MYCALL"** change for connect purposes fooling any user record recorded privilege setting.

To handle non-compliant systems we again go the RFC-822 headers; using "Priority?" buried in the message and passed through other systems recoverable by an RFC-822 saavy system.

b. The TO field should be expanded to at least 32 characters for transmission so that bulletins can be addressed with more intelligence a' la **usenet**. ~~For~~ disk record efficiency, simply defining a total address space of **60-80** bytes and store the **USER@ADDRESS** or **TO-SUBJ@DISTRIB** with the **"@"** embedded in the space. When a bulletin is passed it would have a longer TO field and a shorter DISTRIB field and when a message is passed it has a shorter TO-USER field and a longer ADDRESS field. This

can be implemented transparently **as current TO field parsing for all** systems will truncate and 'correct' the excess without **blowing up. RFC-822 saavy systems would** recover the full TO field

c. The bulletin negotiation currently consists of;

SB TO-SUBJ @ DISTRIB < FROM-CALL \$BID

This is then responded to with an OK/NO. I propose that we **extend the negotiation to** include the title as well

SB TO-SUBJ @ DISTRIB < FROM-CALL \$BID
(title - subject matter - description)

This will permit the receiving system to have more latitude in the determination of what he wants to do with the bulletin. Of course there is no substitute for reading the whole bulletin, but a combination of the title and the remaining information makes an automated disposition decision more refined.

d. Along with the above modification we should universally adopt the 'R' descriptor in the SID from **AA4RE** for extended response which permits us to say RJ to reject a message for as opposed to simply NO which implies its a dupe. We may treat it like a NO, but it will allow us to do more should we care to code it.

In summation, by simple modification of the **"R:"** headers, removal of the continent designators, adoption of the full **ISO-3166** ALPHA-3 list and modification of the forwarding tables removing non-local individual calls, sysops can right now, without software modifications, significantly decrease message sizes and improve message throughput. Further, with some comparatively simple enhancements by the software codewriters, additional significant throughput yields can be realized.

The adoption of the RFC-822 style headers at the head of the main body of the message will provide a standard vehicle for passing message control information across any level system even older non-compliant systems. Because it is an existing international standard it provides compatibility across many messaging systems.

acknowledgements

This paper and its suggestions have grown out of numerous conversations and correspondence with many members of the amateur packet world. I wish to thank J. Gordon **Beattie, N2DSY**, Hank **Oredson**, **WORLI**, Tom Clark, **W3IWI**, Roy Enghausen, **AA4RE**, Tom Moulton, **W2VY**, Frank Warren, **KB4CYC**, for their contributions to the ideas presented here. I would also thank Ralph Stetson, **KDIR**, my sounding board and chief critic while much of the ideas were fleshed out.

brian b. riley, **ka2bqe**
po box 188, **harvey** road
underhill center, **vermont** 05490-o 188

packet:ka2bqe @ w1 koo.vt.usa
cis: 71420,3543
phone: (802) 899-9922 (msg) / (802) 899-4527 (voice)

GUI PACKET

Graphical User Interface On Packet Radio

Keith Sproul, WU2Z
698 Magnolia Road
North Brunswick, NJ 08902
WU2Z @ W2EMU.NJ
AppleLink: Sproul.K

Mark Sproul, KB2ICI
1368 Noah Road
North Brunswick, NJ 08902
KB2ICI @ W2EMU.NJ
AppleLink, Sproul.M

The Radio Amateur Telecommunications Society (RATS)

ABSTRACT

In the 1990's the popularity of Command Line interfaces on personal computers is rapidly fading in favor of Graphical User Interfaces (GUI) on most platforms. A graphical environment using windows, icons and mice has been available on the Macintosh® since '1984. It has also been available on several other computers for many years. Even computer industry insiders were surprised by the headlong rush to Microsoft® Windows by owners of IBM-PCs and clones. Surely the time for packet radio to respond to the overwhelming preference of the majority of computer users is long overdue. It's high time we had a Graphical User Interface that runs over the air-waves! The author has had a packet radio Call Book server working on the air for over a year which can be accessed with normal terminals using off-the-shelf packet software. Users can also connect to it with a GRAPHICAL USER INTERFACE and get access to the same Call Book information, but with a slick GUI instead of the old Command Line interface!

INTRODUCTION

When packet radio started back in the early 80's, most of the people using packet were using 'dumb terminals'. Later on they migrated to computers of some kind running programs to make them look like 'dumb terminals'. Then we started seeing programs such as PC-PakRATT™ and DigiCOM that knew specific things about packet radio. This type of program made the day-to-day activities of packet radio easier. Now we have a version of PC-PakRATT that uses a mouse and we have a Macintosh version called MacRATT™ that takes full advantage of the Mac's graphical user interface. There is also a DOS program called Lan-Link that can do many things for you automatically. These are significant improvements over the 'dumb terminal' programs of the past.

Today, most packet radio operators are using a computer of one kind or another. There are lots of Commodore-64's, IBM-PCs / PC clones and a good number of Macintosh computers and Atari's too. Therefore, since most stations use computers, let's take advantage of this Local Intelligence. The local computer has to do two things: First and foremost, it must take care of all of the *graphics*, i.e. the drawing, icons, and mouse handling. Fortunately the

Macintosh system and MS-Windows take care of all of this. Second and much more important to the designers of the protocols, is that the protocol must be extensible. That is, it must be able to define other primitives within itself like a macro language. This is VERY important to reduce the amount of data that is transmitted over the air.

GRAPHICAL USER INTERFACE OVER PACKET RADIO

The most frequent first reaction I get when I talk about doing graphics over packet radio is that 1200 baud is too slow to be useful when used to transmit the greatly increased formatting overhead that GUI's require, and that doing a graphical user interface over packet radio would not be possible.. Although I agree that 1200 baud is slow and that GUI is not desirable at 1200 baud, it can be made **useable**. I have done it!

Another common comment that I get is that if someone puts up a BBS that supports a GUI type of connection, what becomes of the people with dumb terminals, or computers that can't handle the graphics, or people that just don't want to use a graphical interface? The answer to this is simple; the BBS knows your name and other things about you, it can easily remember what type of interface you want to use. Also, why can't the BBS ask "what are you?", and if it doesn't get the appropriate answer, assume that you have a normal 'dumb terminal'. The system that we have running automatically recognizes whether the station is a normal station or a GUI station.

The **biggest** problem is shaking the packet community out of its complacency. We must demand progress in the area of user interfaces. The second biggest problem is establishing a standard protocol that will work. Once this is done, programs on most computers will become available.

I use Macintosh computers exclusively and have some excellent software for use with packet radio. For connecting to normal PBBSs, I use the program called **MacRATT** from AEA. This program is everything I need for connecting to NORMAL command line PBBSs. It also adds several nice GUI touches that allow me to have an easier interface to PBBS systems that otherwise do not know anything but command line interfaces. This is especially true with the most recent release, **MacRATT** Version 2.1. Although this program is very nice, what I REALLY want is a true Macintosh-like interface, not a command line interface. A true Graphical User Interface will require standardized protocols and local intelligence. This can be accomplished with some existing software such as **MacWorkStation™** or **HyperCard™** on the Macintosh or **Toolbook™** on the PC or it can start from scratch. Due to the variety of computer systems in use, the best way will be to define LOCAL INTELLIGENCE and a DYNAMIC PROTOCOL LANGUAGE.

LOCAL INTELLIGENCE

For an example of local intelligence let's ask, "How many times does a typical user requests on-line help?" If instead of just sending the message up to the PBBS, why doesn't the local packet program check to see what version/date of this particular **PBBS's** help file it has saved. Then ask the current PBBS what version/date is the most recent version, At that point it would ONLY download a new file if it is more recent than the one saved within the program. If the one that is saved is current, then just display it, without re-downloading it every time. That way, the help file would come up on the screen immediately unless there was a new version on the PBBS, in which case, it would take no longer that it does now.

DYNAMIC PROTOCOL LANGUAGE

A dynamic protocol language is a system that can take a long series of commands and call them with a single command. Macros are a simple form of a dynamic protocol language. The important thing that a DYNAMIC protocol language have is the ability to define and/or redefine the commands on the fly. The reason for this is that each different application will probably want to do the same thing over and over. If it can tell the end user **software** that a simple command of only a few letters ACTUALLY means to dlo a couple of pages of commands, then you can greatly reduce the amount of data being transferred over the communications link. Another thing that a dynamic protocol **language** needs is the ability to pass arguments to these 'routines'. A simple example of this **would** be if you had a system that could draw straight lines given x and y coordinates, you would want to be able to define a routine that could draw a square given only two coordinate pairs. Once you have defined this SQUARE routine, you would not have to send nearly as many pieces of information to get your square, and it could be drawn anyplace on the screen and any size.

For examples of a dynamic protocol language, we can look at several different common systems in use today. The most common system out there is POSTSCRIPT? Although mostly recognized as a page description language, Postscript was originally designed as a dynamic protocol language for use on networks. Another **common** example of a dynamic protocol language is FORTH. Forth is a fully functional, interpreted computer language that can be changed on the fly. Both of these systems are described in a little more detail below.

CURRENT IMPLEMENTATION OF A GUI CALL BOOK SERVER

Our current call book server is done via Apple **MacWorkStation** [1] (MWS) software from Apple Computer. There is also a Windows version available **called** ALAC, which stands for Any Language - Any Computer and is available from United Data Corporation, San Francisco, CA. ALAC is a MacWorkStation compatible program that allows the Windows user complete access to the same interface that MacWorkStation gives you.

MacWorkStation is a dynamically changeable system, as defined above. It is not as flexible as Postscript, but it is already defined, and handles a lot more than just drawing on the screen. It already knows about menus, windows, dialog boxes, user events etc. See the sidebar at the end of this paper for a brief introduction to MacWorkStation.

The MacWorkStation package allows programmers on host computers to communicate with remote terminals in 'normal' text-based ways but have the end-user terminal actually display graphics, windows, icons, menus, etc. The initial design goal of MacWorkStation was to allow the main-frame programmer that knew nothing about how to program graphics on the Macintosh an easy mechanism to implement a GUI environment on a remote work station. MacWorkStation went a lot further than the original design goal and has provided a very rich environment. It has provided a very easy way to implement a GUI environment over packet radio. MacWorkStation is only one way that a remote GUI interface can be implemented. It just happened to be a very easy way to reach my goals without having to write a full-function graphics language.

Mark Sproul, KB2ICI, has written a 'Call Book Server' program which enables you to look up Call Book information on anyone with a US FCC issued license. After getting this system running, we wanted to be able to access it via a Graphical User Interface. In **addition** to this we did NOT want to exclude anybody that did not have the capability to use a GUI interface.

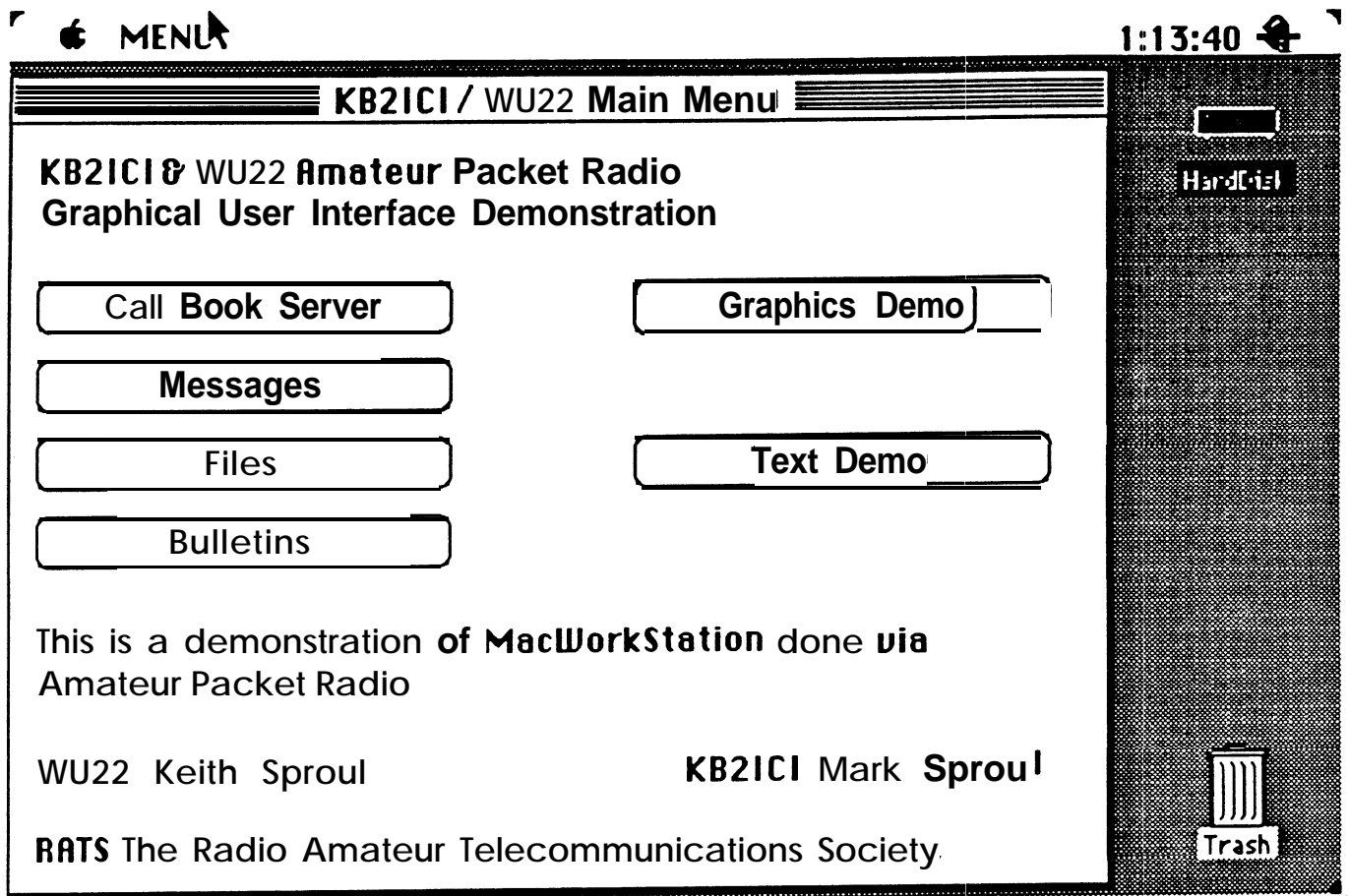
We took the MacWorkStation protocols and added them directly to **KB2ICI's** Call Book **Server** software. This turned out to be easier than first expected due to the fullness and completeness of the MacWorkStation protocols. A couple of screens are shown below in Figures 1 & 2. Figure 1 is the ACTUAL screen you get when you sign onto the **KB2ICI** call book server with the GUI software running on the operator's computer. Listing 1 is the ENTIRE listing of MacWorkStation code needed to generate that first screen. Figure 2 is the Call Book Server screen you get when you push the first button on the main screen. This listing puts up the screen, the code that inserts the actual call book information is not listed here.

The MacWorkStation listings are given to show the quantity of the code and the general nature of the types of commands that are sent. They are not meant to teach you how the MacWorkStation commands work. That information is available in the MacWorkStation manuals. [1] One of the many things that MacWorkStation allows you to do is once these commands have been sent to the remote work station, they can be saved, and called back up later with a simple one line command. This is done in the current implementation of the Call Book Server software. They can also be saved from session to session so that the next time you log on it only takes one command to pull up the entire screen. In this case it will take LESS data to get logged in that even with the minimum command line user interface!

Another thing that MacWorkStation can do very easily is the ability to check for versions and only download the information if it is newer than what is already saved locally. This is the type of thing that was discussed above about local intelligence.

Listing 3 is the same information obtained when you log into the Call Book Server via a normal 'dumb' terminal. In Listing 3, the underlined commands are what the user typed. In response to the W (or WHO) command, the computer gives a listing of who logged in, when, and if they were logged in normally (indicated by '-') or logged in via a Macintosh (indicated by a 'M') The program can also detect if it was an IBM using the IBM Windows program, **ALAC**, that is compatible with **MacWorkStation**.

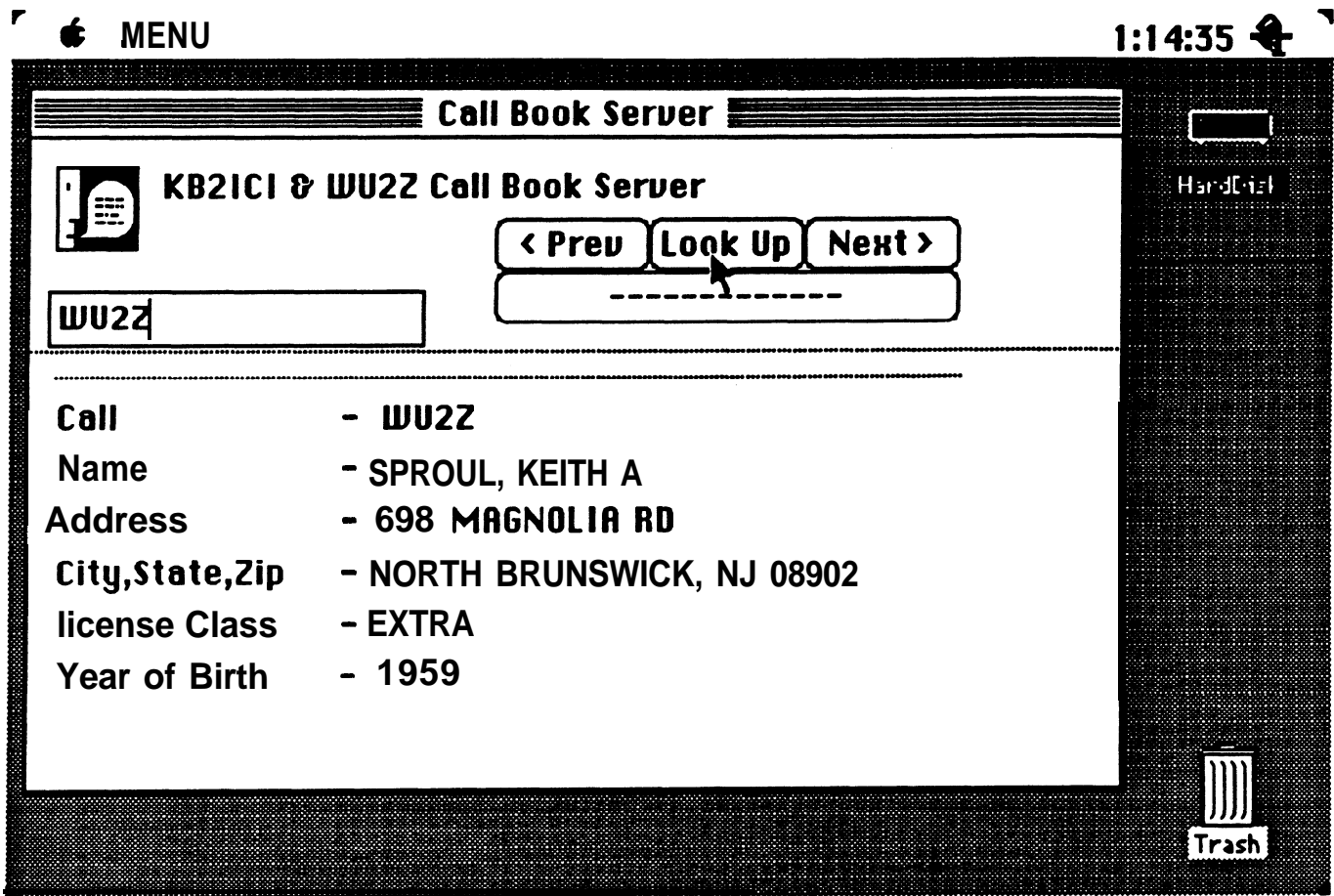
FIGURE 1



Listing 1:

```
[M001 1;  
[M004 1;MENU,5,RETURN TO MAIN MENU,QUIT;  
[M003 1;  
[A001;  
[D001 1; F; KB2ICI / WU22 Main Menu; 4; T; 1,43,422,337;  
[D009 1; B; 10,60,170,80; Call Book Server; T; F; 0;  
[D009 1; B; 10,90,170,110; Messages; T; F; 0;  
[D009 1; B; 10,120,170,140; Files; T; F; 0;  
[D009 1; B; 10,150,170,170; Bulletins; T; F; 0;  
[D009 1; B; 240,60,400,80; Graphics Demo; T; F; 0;  
[D009 1; B; 240,120,400,140; Text Demo; T; F; 0;  
[D009 1; T; 10,10,280,46; KB2ICI & WU22 Amateur Packet Radio Graphical User  
Interface Demonstration;  
[D009 1; T; 10,190,380,226; This is a demonstration of MacWorkStation done via  
Amateur Packet Radio;  
[D009 1; T; 10,240,150,256; WU22 Keith Sproul;  
[D009 1; T; 260,240,400,256; KB2ICI Mark Sproul;  
[D009 1; T; 10,270,400,286; RATS The Radio Amateur Telecommunications  
Society;  
[A001;
```

FIGURE 2



LISTING 2

```
[D008 1;
[D001 2; F; Call Book Server; 4; T; 8,49,430,300;
[D009 2; B; 180,50,360,70; -----; T; F; 0;
[D009 2; B; 180,30,240,50; < Prev; T; F; 0;
[D009 2; B; 240,30,300,50; Look Up; T; F; 0;
[D009 2; B; 300,30,360,50; Next >; T; F; 0;
[D009 2; E; 10,60,150,76; T; 0; T; ; Enter Call Here;
[D009 2; T; 120,100,410,116; -;
[D009 2; T; 120,120,410,136; -;
[D009 2; T; 120,140,410,156; -;
[D009 2; T; 120,160,410,176; -;
[D009 2; T; 120,180,410,196; -;
[D009 2; T; 120,200,410,216; -;
[D009 2; T; 10,100,120,116; Call;
[D009 2; T; 10,120,120,136; Name;
[D009 2; T; 10,140,120,156; Address;
[D009 2; T; 10,160,120,176; City,State,Zip;
[D009 2; T; 10,180,120,196; License Class;
[D009 2; T; 10,200,120,216; Year of Birth;
[D009 2; I; 10,10; 1; T; F; 0;
[D009 2; T; 50,10,310,26; KB2ICI & WU2Z Call Book Server;
[D009 2; L; 360,90; 10,90;
[D009 2; L; 0,80; 420,80;
[A001;
```

LISTING 3

```
01:24:08 CONNECTED to KB2ICI-7
Welcome to the KB2ICI/WU2Z Ham server system
Version 1.03 (C) 1990 by Mark Sproul (KB2ICI)
WU2Z DE KB2ICI-7 @ 20:26 - command (?,B,F,G,H,N,P,W) > F WU2Z
Call entered: WU2Z - searching
      Call: WU2Z
      Name : SPROUL, KEITH A
      Street: 698 MAGNOLIA RD
      City,ST,Zip: NORTH BRUNSWICK, NJ 08902
      Class: E issued in 89353, expires in 99353
      Birth year: 59

WU2Z DE KB2ICI-7 @ 20:26 - command (?,B,F,G,H,N,P,W) > W
System running since 8/ 9/1991 20:19
08-09-1991 20:26 - WU2Z
08-09-1991 29:10 M WU2Z
...
      Number of records in data file: 515325
      Connections made to this server: 2
      Call requests made to this server: 2
WU2Z DE KB2ICI-7 @ 20:26 - command (?,B,F,G,H,N,P,W) > B
73 from the KB2ICI Call Book server
(Note: Underlined characters where typed by the user)
```

GENERAL GUI REQUIREMENTS

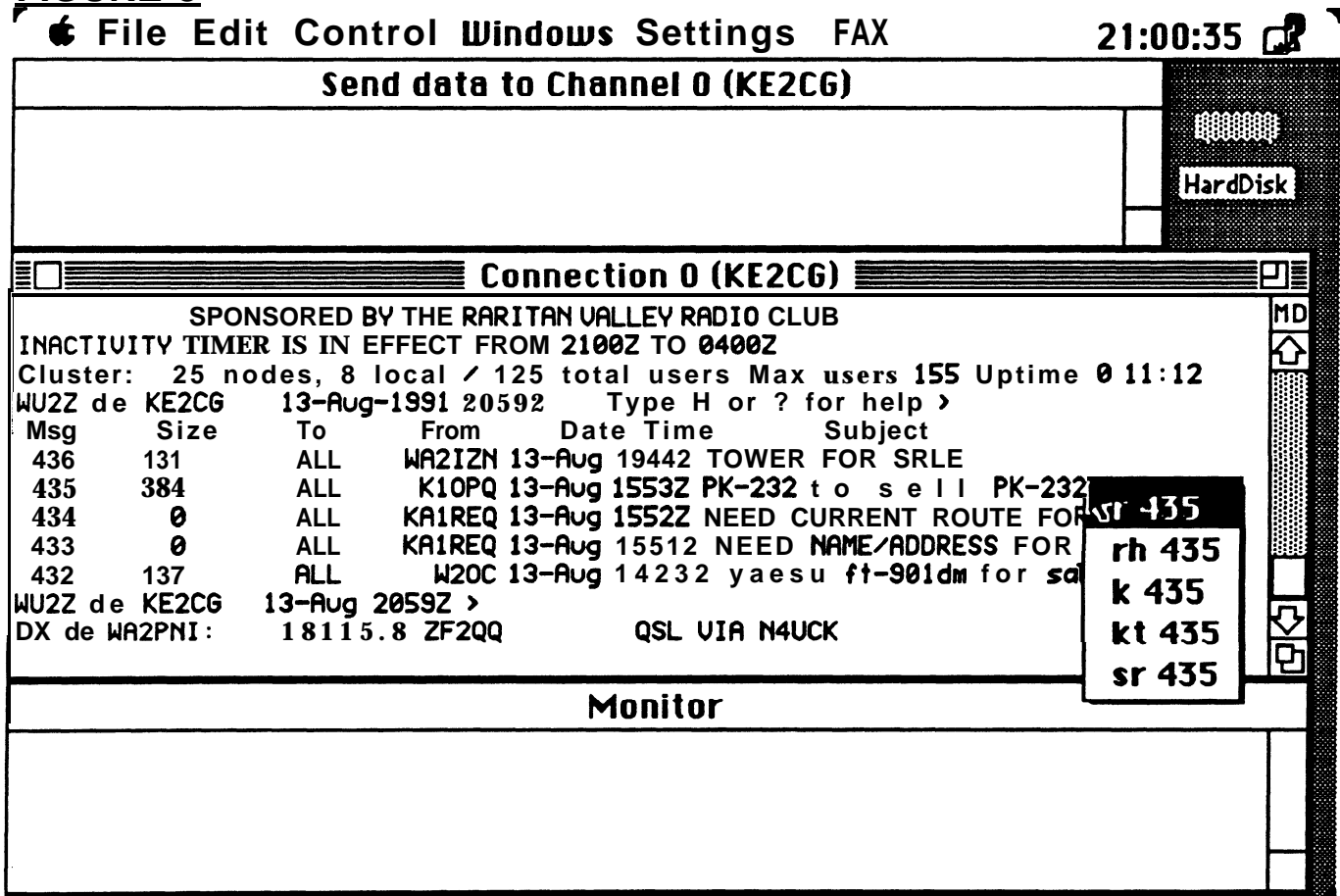
To be able to support graphics of any kind over packet radio, we must set some initial guidelines and some minimum requirements:

- Any system developed should have a “dumb terminal” mode so those users that do not have GUI capability can still obtain useful functionality, and so that all users can connect to the system and establish themselves as GUI capable users on that system. The “dumb” terminal mode would at least be able to provide basic information and help about the system protocols and the ability to register.
- Older computers such as Commodore-64's, CP/M computers, Apple IIs, and even some 8088 and 8086 based PCs must be treated as “dumb” terminals since these computers do not have enough processing power to perform adequately in a graphical system.
- We need a system that is complete yet very flexible.
- We need the ability to most of the things described above in the MacWorkStation description, such as windows, menus, graphics, file I/O, text manipulation, etc.
- The communications protocol must be terse.
- The protocol must be dynamic, i.e. macro type capability.
- It must have the ability to determine what type of environment the end user is using.

PREVIOUS WORK

This type of system has been discussed **before**. [2]. The authors of that paper discussed application specific programs that knew what each other were doing. This type of program is also needed, and fits in to what I call 'local intelligence'. In this case, the local intelligence means that less data has to be communicated because each program knows exactly what to expect from the other. In the specific case cited above, the authors developed a Chess program in HyperCard on the Macintosh. This program new about chess moves, and only had to communicate the actual chess moves over the air-waves., it did not have to transmit any graphics images of any sorts since both ends were running the same program which already had all of the chess pieces and chess board defined. As written, this program could only do chess, but the concept is very good. Any type of multiple user game, graphical or otherwise could be done very easily in this fashion as long as the game was not a fast action game. The other game that comes to mind that would be excellent for this would be Battleship

FIGURE 3



Another example of 'local intelligence' that is already **built** into a packet program is **AEA's MacRATT**. This program 'knows' about the standard way in which **PBBS's** list messages, i.e., every message in the list starts with a line number. **MacRATT** knows this, and if you enable a **QUICKREAD** mode, it will let you click with the mouse **ANYWHERE** on a line, and **MacRATT** will automatically issue the **READ** command for that message. Figure 3 shows how this works. The mouse button is clicked down at the moment this screen show was taken,

showing the pop-up menu with the optional commands. When the mouse button gets released, the selected command gets copied to the send window. The nice touch with this is that the command that gets issued is user selectable from five commands, done via a pop-up menu, and the user can set which five commands to show. This is good because not all PBBS systems use the same commands. This type of 'local intelligence' is only one sided, so it does NOT reduce the amount of data needing to be **communicated**, but it DOES make it a lot easier for the end user. **MacRATT** also has excellent macro capabilities, but so do many of the other packet programs

OTHER POSSIBLE SOFTWARE SYSTEMS

The question then becomes: how do we implement a GUI system over packet **without** having to completely re-write all the software from scratch? We certainly do NOT wish to reinvent the entire wheel, only the rim!

I used the **MacWorkStation** system from Apple Computer. This **allowed** me to achieve my goals quickly. MWS has some advantages and disadvantages. Some of the basic ones are listed below:

Advantage

Already Working
Macintosh & DOS/Windows
Fully defined protocol
Has already been shown to work

Disadvantage

Commercial program
No other platforms without major effort
Possibly over-defined

Other systems could be used. Several of these are discussed briefly below. The problem with any of these others is that a large development effort would be required. For the most part, the alternatives listed below have some or all of the graphics primitives already written. Ideally, we would end up with a system that could be as **complete** as what is available in **MacWorkStation**, and would have the ability to be on more computers than just the Macintosh and DOS/Windows. The problem is getting people to agree and to start doing **something**.

Display Postscript

Though the most widely known use of Postscript is as a "**page** description" or "printer" language, it was in fact originally designed as a network communications language. Postscript does meet all of the requirements for packet GUI application. Postscript has a few limitations: First, it is slow unless you have a very fast computer. Second, it is NOT easy to program. Third, it would be quite difficult to get new versions of Postscript installed on different types of computers.

Display Postscript is available on Macintosh, DOS/Windows, NeXT, **and** UNIX, but most other smaller computers are not fast enough to be able to handle Postscript

Postscript was developed from Forth, it has been extended so much that the two only vaguely resemble each other but the basic structure is still the same.

FORTH

Forth is a very extendable language that is available on ALL known desktop computers. Its use would make it relatively easy to transfer a GUI system from one computer platform to

another. Forth is available on all desktop computers that have had any popularity at all. It has graphics capabilities on any computer that can do graphics.

Forth would be a good choice if you wanted to develop a system for the most types of computers.

HYPERCARD

Apple's HyperCard program and the equivalent DOS/Windows programs (such as Spinnaker PLUS and Asymetrix Toolbook), would be a relatively easy platform to develop a graphical user interface protocol. HyperCard has already been used to do Packet Chess on the Macintosh. [2] Both the Macintosh and DOS/Windows versions already have the graphics primitives done. In the Macintosh environment, this is a program that ships with every Macintosh, but in the DOS/Windows environment, this program is not very widely used yet.

HyperCard is an 'object oriented' system that would be ideal for developing an entire system for this type of high level communications. It also has the ability to be changed dynamically which leads to even more flexible types of systems. HyperCard is the system I would choose to use if I decide to develop an entire system without relying on already developed software such as MacWorkStation.

HyperCard is only available on the Macintosh and DOS/Windows. It would not be feasible to implement on any other computers.

CONCLUSIONS

I would like to see more done for packet radio in the areas of local intelligence, ease of use, and specifically, Graphical User Interfaces. The method I have used is not the only way to implement a GUI system over packet radio. Considering the amount of software that would have had to be written in any other alternative I could think of, MacWorkStation turned out to be the easiest method by far. I will continue to pursue this method and other methods of achieving a Graphical User Interface over packet radio. I would like to see some of the popular BBS systems start to think about adding this type of capability. The author of one of the popular packet BBS packages is currently looking into this type of capability. Hopefully, other people will get interested in this aspect of packet radio and eventually will have more than just the old command line interface.

REFERENCES

- [1] *MacWorkStation Programmer's Guide*, Apple Programmers and Developers Association
- [2] Dewayne Hendricks and Robert Taylor, "Application Software for Packet Radio" in *8th Computer Networking Conference*, Newington, CT, **ARRL**, 1989, pp **210-215** (Colorado Springs, Colorado, October 7, 1989)

SIDEBAR

BRIEF OVERVIEW OF APPLE MACWORKSTATION

MacWorkStation is a **complete** system for allowing host computers to talk to remote terminals using a full Graphical User Interface without having to write any graphics software on the end user computers. It has a very comprehensive command set that covers almost all bases. Listed below are the main **categories** of commands with a brief overview of each.

- **Alert Director** The Alert Director allows the client application to report **error** conditions or otherwise alert the user about something by displaying an alert box. Alerts **range** from a simple beep to a **dialog** box that can ask the user for further instructions, such as to **cancel** the current command or to continue.
- **Cursor Director** The Cursor Director controls the behavior of the cursor on the screen. The cursor's appearance may be changed or the cursor may be hidden. Changing the **cursor's** appearance can be an effective means of communication with the user.
- **Dialog Director** The Dialog Director takes care of displaying and using Dialog boxes on the computer screen.
- **File Director** The File Director controls the exchange of information between the host computer and the end users computer. It allows the host to create files, open and close files, read and write data to files, and get or set information about files.
- **Graphics Director** The Graphics Director handles the creation of graphic images. The host application can draw lines, shapes, text, icons, and complicated pictures in a window by using simple Graphics Director commands.
- **List Director** The List Director allows the host application to display text in a variety of **list** formats. **It** also provides limited editing facilities for the user. Examples of things that can be done with lists would include spread sheet type applications, lists of messages on a **PBBS**, or lists of files that are available for downloading.
- **Menu Director** The Menu Director allows the host computer to tell the MWS to display any menus that are needed by the current application. **It** then only reports back to the host computer when a menu item was actually selected.
- **Process Director** The Process Director handles program control and administration. These tasks include protocol handshaking, ascertaining machine and software version numbers, determining what graphics devices are available, setting user wait states, and exiting the **MacWorkStation** program.
- **Text Director** The Text Director **allows** the user to view, scroll, edit, save, and print text in a text windows.
- **Window Director** The **Window** Director handles the creation and manipulation of different types of windows. There can be **LIST** windows, **TEXT** windows, and **GRAPHICS** windows. The window director also takes care of multiple windows and simultaneous displaying of windows of different types.
- **Exec Director** The Exec Director allows you to extend the **MacWorkStation** application's functionality by adding custom programs of your own to the MWS environment.

As you can see, this is a quite comprehensive list of command groups. Writing a system to handle a full graphics user interface would be quite an undertaking. This particular system has a bit more in it than would be required for a minimum functional system but even writing a minimal system would be a major undertaking. **That** is why the authors choose MacWorkStation.

NOS COMMAND SET REFERENCE

by Ian Wade

G3NRW @ GB7BIL
44.131.5.2

7 Daubeney Close, Harlington,
Dunstable, Bedfordshire,
LU5 6NF, UK

ABSTRACT

This paper contains details of all of the commands to be found in the following KA9Q TCP/IP Network Operating System (NOS) packages:

KA9Q/G1EMM: KH113016 (v1.6) (Nov 1990)
KA9Q/PA0GRI: 910618 (v1.7a) (Jun 1991)

RATIONALISATION OF PARAMETERS

Because the NOS packages contain software modules originating from several different sources, the documentation which describes them inevitably contains a number of inconsistencies. For example, the words `label` and `interface` apparently describe different objects, whereas in actuality they are the same thing. On the other hand, the word `address` can have different meanings, depending on the command.

In this paper an attempt has been made to rationalise the meaning of these parameters, to produce a consistent command set within and across the two NOS packages.

The parameters which often cause confusion are to do with names, addresses and interfaces. These are now defined as follows:

<callsign>	an AX.25 MYCALL callsign (e.g. G3NRW-5)
<hostname>	a host name in DOMAIN.TXT (e.g. g3nrw OR g3nrw.ampr.org.)
<ipaddress>	an Internet address (e.g. 44.131.5.2)
<host>	<hostname> OR <ipaddress>
<username>	a user at a computer (e.g. ian)
<interface>	a device interface name (e.g. pk0)
<ioaddress>	a device I/O base address (e.g. 0x3f8)
<vector>	an IRQ level (e.g. 4)

The word **host id** is not used at all, to avoid confusion with the Unix command of the same name.

KEY

- signifies a command **only in the G1EMM version**
- ¶ signifies a command **only in the PA0GRI version**

Where commands have alternative parameter values, the default value is underscored; e.g. `[off | on]` h e r default parameters are shown in braces; e.g. `{30}`.

?	(help: list of top-level NOS commands)
!	(break out to shell)
#	(comment line)
F10	(escape to NOS command level)

abort	[<session_number>]	(FTP)
-------	--------------------	-------

.....

arp

arp add	<host> ether ax25 netrom arcnet <ether_addr> <callsign>
---------	--

arp drop	<host> ether ax25 netrom arcnet
----------	---------------------------------------

arp flush

arp publish	<host> ether ax25 netrom arcnet <ether_addr> <callsign>
-------------	--

.....

asystat

.....

attach asy	<ioaddress> <vector> slip ax25 nrs ppp <interface> <buffers> <mtu> <speed> [options]
------------	---

option c:	enable RTS/CTS
r:	enable RLSD/CD
v:	enable compression

¶ attach axip	<interface> <mtu> <remote-host> [<callsign>]
---------------	---

■ attach drsi	<ioaddress> <vector> ax25 <interface> <bufsize> <mtu> <chan_a_speed> <chan_b_speed> [<i paddress_a>] [<i paddress_b>]
---------------	--

■ attach eagle	<ioaddress> <vector> ax25 <interface> <buffers> <mtu> <speed> [<i paddress_a>] [<i paddress_b>]
----------------	--

■ attach hapn <ioaddress> <vector> ax25
 <interface> <rx bufsize> <mtu>
 csma | full [<ipaddress>]

■ attach hs <ioaddress> <vector> ax25
 <interface> <buffers> <mtu>
 <txdelay> <persistence>
 [<ipaddress_a>] [<ipaddress_b>]

attach kiss <asy interface> <port> <interface>
 [<mtu>]

attach netrom

attach packet <vector> <interface>
 <tx_queue_length> <mtu>
 [<ipaddress>]

■ attach pc100 <ioaddress> <vector> ax25
 <interface> <buffers> <mtu>
 <speed>
 [<ipaddress_a>] [<ipaddress_b>]

attach scc <devices> init <ioaddress>
 <spacing> <Aoff> <Boff>
 <Dataoff> <intack> <vector>
 [p] <clock> [hdwe] [<param>]

attach scc <chan> slip|kiss|nrs|ax25
 <interface> <mtu> <speed>
 <bufsize> [<cal lsign>]

.....

attended [off|on]

ax25 bc <interface>
 ax25 bcinterval [<seconds>] {0}
 ax25 bctext ["<broadcast_text>"]
 ax25 btimit [<val>] {30}
 ax25 digipeat Coff|on
 ax25 flush
 ax25 heard
 ax25 irtt [<milliseconds>] {5000}
 ax25 kick <&AXB>
 ax25 maxframe [<window_size>] {1}
 ax25 mycall [<callsign>]
 ax25 paclen [<bytes>] {256}
 ax25 pthresh [<bytes>] {128}
 ax25 reset <&AXB>
 ax25 retry [<n>] {10}
 ax25 route
 ax25 route add <target callsign>
 [<digi_callsign> ...]
 ax25 route drop <target_callsign>
 ax25 route mode <target_callsign>
 [vc|datagram|interface]
 ax25 status [<&AXB>]
 ax25 t3 [<milliseconds>] {0}
 ax25 t4 [<seconds>] {300}
 ax25 timertype [original | linear|exponential]
 ax25 version [1|2]
 ax25 window [<bytes>] {2048}

bbs

Help ? (command list)
 Area A [<area_name>]
 Bye B
 Chat C
 Download D [<filename>]
 Escape E [<esc_char>] (^X)
 Finger F [<username>] [<@host>]
 Gateway G <interface> <cal lsign>
 [<digi_callsign>...]
 Help H [<command_letter>]
 Info I
 Heard J
 Kill K <n> ...
 List L [<n> ...]
 Netrom N
 Read R <n> ...

Send S <username> [%<host>] [<@host>]
 [< <from_addr>] [\$<bulletin_id>]

Forward S F <username> [%<host>] [<@host>]
 [< <from_addr>] [\$<bulletin_id>]

Reply SR [<n>]

Telnet T <host> [wel l_known_port_number] {23}
 Upload U <filename>
 Verbose V <n> ...
 What U [<directory>]
 Zap Z <filename>

Remote @
 Expert [<string>]
 Next Message "
 (unknown) #

cd [<directory>]
 close [<session_number>]
 comm <interface> <string>
 connect <interface> <callsign>
 [<digi_callsign> ...]

delete <filename>
 detach <interface>

¶ dialer <interface> [<file>[<seconds>
 [<pings>[<host>]]]]

• dialer <interface> <seconds> <target-host>
 <dialer filename>

■ dialer <interface> 0 (turns dialer off)

```

dir          [<directory>|<filename>]

di sconnec t [<session_number>]                (AX.25)
.....

domain addserver <host> [<host> ...]

domain cache clean [off|on]
domain cache list
domain cache size [<entries>]                  {20}
domain cache wait [<seconds>]                  {300}

domain dropserver <host> [<host> ...]
domain list
domain maxwait [<seconds>]                      {60}
domain retry [<n>]                              {2}
domain suffix [<domain_suffix>]                { .ampr.org. }
domain trace [off|on]
domain translate [off|on]
domain verbose [off|on]
.....

• drsistat

dump <hex memory-address> | <. > [<decimal_range>]

      (memory address is 8 hex chars without colon)

```

```

• eaglestat

echo [accept|refuse]                            (telnet)

eol [standard|null]                             (telnet)

escape <literal-character>                      (also F10 on PC)

exit

```

```

finger [<username>] @<host>
      (no spaces between parameters)
.....

```

¶ fkey

```

¶ fkey <key_number> [<value> | "<string>" I
      (use ^M for CR)

```

f1	59	sf1	84	cf1	94	af1	104	pgup	7 3
f2	60	sf2	85	cf2	95	af2	105	pgdn	81
f3	61	sf3	86	cf3	96	af3	106	home	71
f4	62	sf4	87	cf4	97	af4	107	end	79
f5	63	sf5	88	cf5	98	af5	108	arup	72
f6	64	sf6	89	cf6	99	af6	109	ardn	80
f7	65	sf7	90	cf7	100	af7	110	ar l	75
f8	66	sf8	91	cf8	101	af8	111	ar r	77
f9	67	sf9	92	cf9	102	af9	112	ins	82
		sf10	93	cf10	103	af10	113	del	83

.....

```

ftp <host>

asci i
batch [off|on]
binary
cd <remote_dir>
dele <remote_file>

dir [<remote_dir> | <remote_file>
      [<local_file>] I

get <remote_file> [<local_file>]
hash

list [<remote_dir> | <remote_file>
      [<local_file>] I

ls [<remote_dir> | <remote_file>
      [<local_file>] I

mget <remote_file> [<remote_file> ...]
mkdir <remote_dir>
mput <local file> [<local file> ...]

nlst [<remote_dir> | <remote_file>
      [<local_file>]]

pass <password>
put <local-file> [<remote_file>]
pwd
quit
rmdir <remote_dir>
type [a | i | l <bytesize> I
user <username>
verbose [<n>]                                n=0: errors only
                                           1: + summary
                                           2: + progress
                                           3: + hash
.....

ftype [asci i | binary]

F10                                (to escape to NOS command level)

```

■ hapnstat

```

help                                (list of top-level NOS commands)
.....

```

```

hop check <host>
hop maxtt l [<hops>]                  {30}
hop maxwait [<seconds>]              {5}
hop queries [<count>]                {3}
hop trace [off|on]
.....
hostname [<mail box name>]

```

■ hs

```

icmp echo [off|on]
      (must be on for one-shot ping)

icmp status
icmp trace [off|on]                  (turn off for hop check)

```

```

.....
ifconf ig [<interface>]
ifconfig <interface> broadcast &cast ipaddress>

ifconfig <interface> encapsulation
                                none|ax25|slip|netrom

ifconfig <interface> forward    <fwd interface>
ifconfig <interface> ipaddress  <ipaddress>
ifconfig <interface> linkaddress
                                <callsign|enet addr>

ifconfig <interface> mtu        <bytes>
ifconfig <interface> netmask    [Ox]<hexmask>
ifconfig <interface> rxbuf
.....
¶ info
.....
ip address [<ipaddress>|<hostname>]
ip rtimer [<seconds>]           {30}
ip status
ip ttl    [<hops>]              {255}
.....a.....
isat      [off|on 3]

kick      [<session_number>]

log       [<log_filename>|stop]

mail
.....
mbox
nbox attend    [off|on]
mbox kick
mbox maxmsg    [<n>]             {200}
mbox motd      ["<string>"]
mbox status
mbox timer     [<seconds>]       {0}
mbox t i p t imeout [<seconds>]  {180}
.....
¶ mem circular    [off|on]
mem efficient     [off|on]
mem free
mem garbage
mem ifbufsize    [<bytes>]       {2048}
mem nibufs       [<n>]           {5}
mem sizes
mem status
mem thresh       [<bytes>]       {8192}
.....
mkdir <di rectory>

mode <interface> [vc|datagram]   {AX.25}

more <filename> [<filename> . . .]
                                (q: quit)
                                (space: next page)
                                (CR: next line)

```

```

motd      ["<string>"]
rnultitask [off |on]

netrom acktime    [<milliseconds>]      {3000}
netrom bcnodes    <interface>
netrom connect    <node_callsign>|<node_alias>
netrom choket ime [<milliseconds>]      {180000}
netrom derate     [off |on]
netrom interface  <interface> <alias> <quality>
netrom irtt       [<milliseconds>]      {15000}
netrom kick       <&CB>
¶ netrom load      <filename>
netrom minqual i ty [<n>]                {10}

netrom nodefilter
netrom nodefi lter add <neighbour_callsign>
                                <interface>
netrom nodefilter drop <neighbour_callsign>
                                <interface>
netrom nodefilter mode [none|accept|reject]

netrom nodetimer  [<seconds>]            {0}
netrom obsotimer  [<seconds>]            {0}
netrom promiscuous [off|on3]
netrom ql imi t   [<bytes>]              {2048}
netrom reset      <&CB>
netrom retries    [<n>]                  {10}

netrom route
netrom route add  <alias> <destination> <interface>
                                <quality> <neighbour>
netrom route drop <destination> <neighbour>
                                <interface>
netrom route info <destination>

¶ netrom save      <filename>
netrom status
netrom timertype   [l i near|exponent: i a l]
netrom ttl         [<hops>]              {10}
netrom user        [<username>]
netrom verbose     [off |on]
netrom window      [<frames>]            {4}
.....
nntp addserver    <nntpserver host:>
                                [<interval_in_seconds>]
                                [<time_range>]
                                [<group> [<group> . . .] l]

nntp directory    [spool | cont rol • :di rectory>]
nntp dropserver   <nntpserver>
nntp groups       [<newsgroup_name> . . . ]
nntp kick         <nntpserver>
nntp listservers
nntp trace        [<n>]                  n=0: no trace
                                1: serious errors
                                2: transient errors
                                3: session progress
                                4: received articles
                                5: errors
.....

```

```

nrstat
-----
¶ parsm <interface>
  param <interface> <param> [<param> . . .]

  parsm <KISS_interface> 0 <data frame>
  param <KISS-interface> 1 <TX_delay>      (10mS units)
  param <KISS-interface> 2 <persistence>    (0-255)
  param <KISS_interface> 3 <slot_time>     (10mS units)
  param <KISS_interface> 4 <TX_tail>       (10mS units)
  param <KISS-interface> 5 <n>             (n=0: HDX)
                                         (n>0: FDX)

  param <KISS_interface> 255               (exit KISS)
  .....

ping <host> [<len> [<seconds> [<incf lag>]]]
.....

pop mailbox <mbox>
Pop mailhost [<host>]
pop kick
pop quiet    [off|on]
pop timer    [<seconds>]                      {0}
pop userdata [<username> <password>]
.....***

ps
pwd      [<directory>]
-----

¶ rarp
¶ rarp query <interface> <ether_addr> | <callsign>
              [<ether_addr> | <callsign>]

record      [<filename>|off]

remote      [-p <port>] [-k <key>]
              [-a <kickaddr>] <host>
              exit|reset|kick

remote      -s <key>

rename      <old filename> <new_filename>

reset       [<session number>]
.....

rip accept  <incoming gateway-host>

rip add     <destination-host> <secs> [<flags>]
              (1: include route to self)
              (2: split horizon)
              (4: triggered update)

rip drop    <destination-host>
rip merge   [off|on]
rip refuse  <incoming_gateway-host>
rip request <incoming_gateway-host>
rip status
rip trace   [<n>]          n=0: no trace
                          1: changes only
                          2: full trace
.....

```

```

¶ rlogin    <host>
rmdir      directory,
.....

route
route add   <dest_host> [/<bits>] | default
              <interface>
              [gateway-host [<metric>]]

route addprivate <dest_host> [/<bits>] | default
              <interface>
              [gateway-host [<metric>]]

route drop  <dest host> [/<bits>]
route flush
route lookup <dest_host>
.....

rspf interface [<name> <quality> <horizon>]
rspf message   [<@message_string">]
rspf maxping   [<n>]                      {5}
rspf mode      [vc|datagram|none]
rspf rrhtimer  [<seconds>]                {0}
rspf routes
rspf status
rspf suspecttimer [<seconds>]
rspf timer       [<seconds>]              {0}
-----

sccstat
session [<session_number>]
shell
.....

smtp batch    [off|on]
smtp gateway  [<host>]
smtp mode     [queue|route]
smtp kick
smtp kill     <job number>
smtp list
smtp maxclients [<n>]                      {10}
smtp quiet    [on]
smtp timer    [<seconds>]                  {0}
smtp trace    [<n>]                        n=0: trace off
                                          1: trace on

smtp usemx    [off|on]
.....

socket       [<socket_number>]

source       <script_filename>

start        ax25|discard|echo|finger|ftp|netrom|pop|
              remote|rip|smtp|telnet|ttypink

start tip    <sync_interface>

status

stop         ax25|discard|echo|finger|ftp|netrom|pop|
              remote|rip|smtp|telnet|ttypink

stop tip     <sync interface>
-----

```

```
tail <filename>

.....
tcp irtt      [<millisecs>]          {5000}
tcp kick      <&TCB>
tcp mss       [<bytes>]              {512}
tcp reset     <&TCB>
tcp rtt       <&TCB> <millisecs>
tcp status    [<&TCB>]
tcp syndata   [off]
tcp timertype [linear|exponential]
tcp trace     [off]
tcp window    [<bytes>]              {2048}
.....

telnet <host>  [<well_known_port_number>] {23}
telnet <host> 87 (CHAT/TTYLINK)
```

¶ test

```
third-party  [off|on]
```

```
tip          <async interface>
```

¶ **ttylink** <host> [<well known port_number>] {87}

¶ **trace**

```
trace <interface> [<BTIO flags>
                  [<trace-filename>]]
```

BTIO_flags:

```
B=0 Broadcast filter off (trace all packets)
B=1 Broadcast filter on (ignore broadcasts)
```

```
T=0 Display protocol headers only
T=1 Display headers + ASCII text
T=2 Display headers + ASCII text + hex
```

```
I=0 Ignore input packets
I=1 Trace input packets
```

```
O=0 Ignore output packets
O=1 Trace output packets
```

```
udp          status
```

¶ **upload**

```
upload <filename>
```

```
watch        [off|on]
```

```
watchdog    [off|on]
```

NOS STARTUP OPTIONS

```
nos [-b] (console BIOS)
      [-d <root_directory>]
      [-m <heap_memory_in_KB>]
      [-s <socket_array_size>]

      [ <nos autoexec-filename> ]
```

FTPUSERS PERMISSIONS

ftp and telnet

```
1 read file
2 create new file
4 write/delete file
```

telnet only

```
8 AX.25 Gateway access
16 Telnet Gateway access
32 NET/ROM Access
64 Remote control
128 Disallow access
```

WELL-KNOWN PORT NUMBERS

0 reserved	23 TELNET	79 FINGER
1-4 unassigned	25 SMTP	87 TTYLINK
5 RJE	37 TIME	95 SUPDUP
7 ECHO	39 RLP	101 HOSTNAME
9 DISCARD	42 NAMESERVER	102 ISO-TSAP
11 USERS	43 NICNAME	109 POP-2
13 DAYTIME	53 DOMAIN	113 AUTH
15 NETSTAT	67 BOOTPS	117 UUCP-PATH
17 QUOTE	68 BOOTPC	119 NNTP
19 CHARGEN	69 TFTP	513 RLOGIN
20 FTP-data	75 private dialout	
21 FTP	77 private rje	

Copyright (c) 1990, 1991 Ian Wade, G3NRW. All rights reserved. No part of this paper may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system for commercial purposes or resale or barter without written permission from the author.

This paper may be reproduced in whole or in part for any non-commercial amateur radio purpose provided the author is credited.

17 July 1991

Higher Speed Amateur Packet Radio Using The Apple Macintosh Computer

by
Doug Yuill VE3OCU, Ottawa, Canada
dyuill@ccs.carleton.ca

Abstract

This paper will discuss higher speed amateur packet radio experiments conducted with the Apple Macintosh computer and the **WA4DSY** radio modem. In particular, the results of using two different hardware/software combinations will be explored. The first combination uses **KA9Q's** NET/Mac program with a modified Pat-Comm TNC-2; the second combines Intercons **TCP/Connect II™** with the Gracillis **PackeTen** NOS in a box, a stand-alone IP switch.

Introduction

Packeteers in the Canadian National Capital region have enjoyed **higher** quality networking through the use of a high-speed, wide area, digital repeater and packet switch known as Hydra [1]. This repeater has served as the focal point for experimentation with advanced computer-mediated communications (CMC). Access to the resources of a Unix based university size network has created the need for better end-user software. Phil Kam **KA9Q's** **TCP/IP** software NET/Mac provides robust access to a core set of **tcp/ip** applications [2]. In conjunction with the Gracillis **PackeTen** [3], the Macintosh becomes a platform, rich in CMC possibilities. Commercial software such as Intercon's **TCP/Connect II™** can be connected via a SLIP (serial line **internet** protocol) interface to the **PackeTen**. This configuration has the dual advantage of providing an improved quality user interface with support for NNTP (network news transport protocol), SNMP (simple network management protocol) & POP (post office protocol), while still supporting AX25 via **KA9Q's** more advanced NOS (network operating system), running on the Motorola 68302 in the **PackeTen**. The **PackeTen** also serves to filter out unwanted packets which would otherwise reduce the performance of the Mac.

History

In the spring of 1989 four members of the Ottawa Amateur Radio Club's Packet Radio Working Group each decided to purchase WA4DSY radio modem kits [4]. This decision was brought about by dissatisfaction with attempts to utilise "off the shelf" voice grade radios to achieve 9600 baud packet radio operation with Steve Goode K9NG's FSK modem [5]. The narrow bandwidth of such radios makes it difficult to pass data at 9600 baud. Also, pre-emphasis and de-emphasis must be used to minimise the group delay caused by non-linear IF stages present in many voice grade rigs. This has the effect of decreasing the inter-operability between different manufacturers' radios. The WA4DSY was seen as a means to circumvent these problems. The extra cost of the WA4DSY modem and transverter was easily justified by the 6 fold increase in performance (56k baud). Off the shelf transverters were used to convert the 28Mhz I.F. of the WA4DSY modem. to 220Mhz where the Canadian D.O.C. has allocated frequency spectrum for experimenting with wider bandwidth packet radio in the form of two 100Khz wide channels.

While the WA4DSY modem provides a stock solution for higher speed packeteering from the RF end of things, its high data rate required a more creative solution for interfacing it to a host computer. Documentation provided with the modem describes, in step-by-step detail, how to modify a standard TNC-2 for use as a host interface [6]. This includes replacing the TNC's firmware with a special high-speed version of KISS. As the WA4DSY modem is of a synchronous design, the KISS firmware assembles the packets so that they can be passed asynchronously to the host computer for further processing. This type of interface was used for the initial testing of the modems on the air. Unfortunately, 56k baud is too fast a data rate for most computers using interrupt-driven serial I/O. This requires the TNC to buffer the packets and exchange them with the host at a lower data rate, usually 19.2k baud. Of course this reduces the maximum effective speed by a full two thirds. On the Macintosh this had the dubious advantage of hiding some performance bottle-necks which will be described in detail later. After several months of successful testing using half duplex links between home QTHs, it was realised that a repeater could provide continuous coverage to a growing higher speed user community while also providing a mechanism for overcoming the dreaded hidden terminal syndrome [7]. A cross-band digital repeater was designed and built for about \$1000 using mostly off the shelf hardware (220.55 Mhz input, 433.55 Mhz output). The bit re-generator circuit designed by Barry McLarnon VE3JF is the only custom part of the design. The repeater has been continuously operational from the Dunton tower at Carleton University since January of 1990.

The First Configuration

I was inspired to finish constructing my modem for testing with my Macintosh, after Marcus Leech (**VE3MDL**) and Barry **McLarnon (VE3JF)** made the first successful radio contacts using the **WA4DSY** modem over a 15 km. path in August of 1989. I installed and configured **KA9Q's** NET/Mac software for use with a Pat-Comm Tiny-2 TNC running KISS-56k firmware. The host interface was set to run at **19.2k** baud.

After noting a lower amount of RF output from the transmit oscillator of the modem than was suggested in the alignment procedure, and being unable to increase it, I decided to try an on-air test over a 20km metro-wide path. With seven-element yagis at each end using lo-watt **transverters** on **220.55Mhz**, I achieved a 100% ping return rate for hours at a time over thousands of packets. During 4 months of testing, the quality varied between a 0% ping return rate for up to 2 days at a time (on several occasions) and the more usual **70%-80%** [8]. The wider bandwidth used by the **WA4DSY** modem probably requires a higher margin for the link budget.

The rtt (round trip time) averaged about 300ms after installing a **bandpass** filter in front of the receive converter to stop intermittent falsing of the DCD (data carrier detect). Performance in file transfers left much to be desired with only hundreds of bytes per second being moved. In part, this was due to the slow interrupt driven I/O of the PC's at the other end of the connection. As well, smaller than necessary values for tcp mss (maximum segment size) were causing the txd (transmit delay timer) to play a much bigger role in overall efficiency. Also, the **19.2k** baud rate between the TNC and computer was creating a longer rtt. Be that as it may, I considered what changes could be made to the TNC to accommodate a higher baud rate. I discovered that by tripling the frequency of the CPU clock to 14.7456 Mhz and then jumpering the CPU for half-speed operation (7.3728 Mhz), I could achieve a **57.6k** baud rate on the host port, the maximum asynchronous data rate available on the Macintosh using NET/Mac. The higher clock speed of the Z-80 would help the KISS-56k firmware service the now three times faster host port.

After replacing the CPU and SIO chips in the TNC with parts rated for 6Mhz operation, upgrading the 4.9152 Mhz crystal to 14.7456 Mhz and making the additional Printed Circuit Board trace cuts and jumpers for higher speed operation, I was ready for more testing. The modifications worked fine and I was now the fastest user on our network! It was noted that the channel loading seemed evenly distributed with four regular users and the Mac did not pause to service I/O from the TNC. This condition was not to last much longer.

Without any other higher speed users to test with, I was still unable to measure maximum performance. This did not last long as Dave Perry VE3IFB started testing the driver software for a high speed DMA (Direct Memory Access) interface board he had prototyped for use on the IBM PC [9]. File transfers were now measured at 3k bytes per second during an ftp get using an IBM PC as a server. Rtt's of less than 100ms were now common. It was observed that this transfer rate was still less than 1/2 of the theoretical maximum. I determined that this was the result of the TNC running in "stop & wait mode". In this mode the TNC can only service either the WA4DSY modem port or the Mac port, but not both simultaneously. The honeymoon created by this ten-fold increase in performance ended abruptly when the second PI board came on-line. The (soon to be infamous) "flatlining" problem now began to occur [10]. This situation was created during ftp sessions between the two PI board users during which almost continuous DCD was observed. This created a band-width bottleneck between the TNC and the Mac in which the Mac (Mac+ with 25Mhz 68030) would stop dead for up to five minutes at a time while it serviced interrupts from the TNC. When this first started happening I thought my machine had crashed! A solution was clearly required.

The Second Configuration

After discovering the flatlining problem, I realised I needed a packet switch to filter out all of the traffic that was not directed to me. It was only logical that with new hardware should come newer software. I evaluated Intercon's TCP/Connect II™ v1.0.1 & v1.0.7. Version 1.0.1 provides a debug window which proved to be quite useful for monitoring system performance. Unfortunately, it is not present in version 1.0.7. TCP/Connect utilises POP (post-office protocol) for supporting E-mail. After several attempts to test Intercon's POP client with NOS, the results are still inconclusive. It should be noted that the KA9Q SMTP (simple mail transfer protocol) implementation in NET/Mac worked very well. A new graphical mail browser written by R. Taylor KA6NAN provides an excellent user interface. NET/Mac's lack of support for cut, copy and paste between session windows (except for mail) and the absence of standard terminal emulation with cursor positioning has limited its usefulness with Unix. Memory usage by NET/Mac is 600k bytes under Multi-Finder vs. TCP/Connects 1.8 Mbytes. These heap sizes were found to prevent system crashes during continuous operation of days at a time while allowing a complete exercise of features and facilities. TCP/Connect's support for NNTP is superlative. The client can be customised for font size and type in all three panes of the browser window. It's even possible to cause the subject field of the message text to appear in bold!

Tests were conducted with a server running on a Sun-4 with subscriptions available to over 1200 different groups. The ftp client is also graphical and provides an intelligent interface to its service. The configuration palette provides a quick and intuitive setup. Telnet support provides a complete set of terminal emulations for DEC, Tektronics and IBM. Finger is supported via a server only. In all cases except **SMTP**, where **TCP/Connect** did not provide support for a service, the **PackeTen** did. The mailbox feature of the **PackeTen** is accessible via telnet with an escape to the NOS console supported for remote administration. The **PackeTen** firmware was found to be of adequate quality. A problem setting the SLIP data rate at any speed other than 9600 baud, is expected to be fixed in the next release of the firmware. During the initial (mis)configuration of the **PackeTen**'s 2k EERPOM some crashes were experienced. The following configuration information has run the **PackeTen** for weeks with-out a crash:

```

Hostname: switch.ve3ocu.ampr.org
Site Alias Name: OCU-SW
IP address: 44.135.96.13
AX25 mycall: VE3OCU
attach asy 0 slip sl0 2048 1536 57600 [44.135.96.13]
route add [44.135.96.12] sl0
Sysop password: NotReally
Additional Command List:
0: attach sync302 1 hdx ax25 ax0 2048 2048 0 ext ext
[44.135.96.13]
1: route add [134.117]/16 ax0 44.135.96.35
2: route add default ax0
3: ifconfig ax0 mtu 1536
4: param ax0 0 15
5: start telnet
6: attach netrom ; start netrom ; netrom interface ax0
600C 190
7: start tty ; domain addserver 134.117.1.1
8:
9:
10:
11:

```

Cost/benefit

The cost of the DSY Modem is minimal when compared to the increase in performance. By investing more in equipment, real-time, high performance advanced CMC applications will become possible. This might include multi-media E-mail complete with full motion video. Or even more exciting, Virtual Reality experiments. Remember, the WA4DSY modem is already 10 times less expensive than a typical 1200 baud packet station if you factor in performance. About \$700 for 3k bytes/second vs. \$230 for 100 bytes/second. Of course, the PackeTen and any additional software will increase the cost beyond that of a basic station. Why be “penny wise and pound foolish” when the price/performance ratio of higher speed Packeteering makes the extra cost of upgrading so palatable?

Future Directions

With the addition of an Ethernet interface for the PackeTen, the SLIP interface could be discarded. This would then completely integrate all of the native Macintosh networking resources available through Apple's Communications Tool Box while still supporting AX25 in the PackeTen. Such a configuration could provide distributed file system experimentation. Support is already available through Intercon's NFS/Share™, a Macintosh NFS (Network File System) client. Also, either White Pine Software or Apple's X-window server could be used to gain access to any X clients such as the XRN newsreader or XMH, a graphical version of the Rand Mail Handler. It should also be possible using Multi-Finder to simultaneously run the Macintosh version of KA9Q's NET via a SLIP interface to the PackeTen. This would provide SMTP service albeit, via a separate IP address. This would still leave the second serial port available on the Mac for experimenting with Timbuktu/Remote via the console port of the PackeTen. Timbuktu/Remote is a utility that allows complete remote control of a Macintosh. It is expected that the presently available software will create an acute need for the extra speed afforded by the inexpensive digital microwave transceiver technology now under development [11].

Conclusions

“...everytime you make an amplifier for a deep human need, you have a winner regardless of whether it has ever been done before...” Alan C. Kay [12]

Warren Toomey VK1XWT [13] has suggested that “...papers involving digital radio communications may be useful in swaying the minds of some of the Universities who control Internet access.”, noting also that these papers might “place amateur packet activity on a research footing”. Based on the results of my experience with both the amateur radio and University communities, fostering such cooperation is a real possibility. The challenge for builders of amateur packet radio networks is to **realise** a fully connected class A internet capable of maintaining a minimum quality of service. This should involve an organised effort to interconnect more users of the .44 internet address space.

I hope I have suggested the scope of creative and cooperative projects that can be undertaken on a network of inter-operatable hosts and clients, distributed between amateur and academic computer networks.

Special thanks to all the members of the Ottawa Amateur Radio Clubs' Packet Working Group including Dr. Warren Thorngate & Dr. Frances Cherry. Without them, this research would not have been possible.

References

1. McLarnon, Barry: Notes on a Regenerating Repeater Using the WA4DSY Modem (first distributed at the T.A.P.R. General Meeting in Tucson, February 1990). Also see Barry McLarnon VE3JF's paper elsewhere in these proceedings for a more complete description of the Ottawa amateur packet radio network.
2. Hendricks, Dewayne WA8DZP and Thorn, Doug N6OYU. “Status report on the KA9Q Internet protocol package for the Apple Macintosh”. Proceedings of the 9th Computer Networking Conference. London ONT: ARRL, 1990, pp. 118-121.
3. Lemley, Don N4PCR and Heath, Milt. “The PackeTen® System: The Next Generation Packet Switch”. Proceedings of the 9th Computer Networking Conference. London ONT: ARRL, 1990, pp. 170-176.
4. Heatherington, D. WA4DSY. “A 56 Kilobaud RF Modem”. Proceedings of the 6th Computer Networking Conference, 1987.
5. Goode, Steve “Modifying the Hamtronics EM-5 for 9600 BPS Packet Operation”, Proceedings of the 4th Computer Networking Conference, 1985.

6. Modifications to the TAPR TNC-2 for 56 KB service (2/24/88) in Heatherington 56 KBRF Modem, Release 1.0 Notes, Revision 4/22/88. Distributed through Georgia Radio Amateur Packet Enthusiasts Society, Inc., 56 KB Modems, PO Box 871, Alpharetta, GA 302339-0871.
7. Avent, Scott and Finch, Robert "A Duplex Packet Radio Repeater Approach To Layer One Efficiency Part Two". Proceedings of the 7th Computer Networking Conference, 1988. Columbia, Maryland pp. 1-9.
8. Drinnan, David J. VE3MPX "The Hand of God: I Feel the: Ghost in the Machine." (in progress)
9. Perry, Dave VE3IFB "The Ottawa Packet Interface (PI): A Synchronous PC Interface for Medium Speed Packet Radio", elsewhere in these Proceedings.
10. Garbee, Bdale N3EUA Tcp-group internet communications.
11. Elmore, G. N6GN and Rowett, K. N6RCE "Inexpensive Multi-megabaud microwave data link." Ham Radio, 1989 Dec., 9-29.
12. Kay, Alan Curtis, "Learning vs. Teaching with Educational Technologies", EDUCOM Bulletin, 1983, Fall/Winter, 16-20.
13. Toomey, Warren. VK1XWT "References for papers on. digital communications research in the tcp-group internet mailing list of 2 Aug. 91".

NOTES

NOTES

NOTES